Universidad de Alcalá
Escuela Politécnica Superior

**Grado en Ingeniería en Tecnologías de la Telecomunicación**

**Trabajo Fin de Grado**

# Demostrador para la detección de personas y el análisis de su actividad en imágenes de color.



**Autor:** Juan Manuel Vera Díaz

**Tutor:** Cristina Losada Gutiérrez

2017

# UNIVERSIDAD DE ALCALÁ

ESCUELA POLITÉCNICA SUPERIOR

**Grado en Ingeniería en Tecnologías de la Telecomunicación**

**Trabajo Fin de Grado**

**Demostrador para la detección de personas y el análisis de su actividad en imágenes de color.**

Autor: Juan Manuel Vera Díaz

Tutor: Cristina Losada Gutiérrez

**Tribunal:**

**Presidente:** Alfredo Gardel Vicente

**Vocal 1º:** Carlos Andrés Luna Vázquez

**Vocal 2º:** Cristina Losada Gutiérrez

Calificación: .....

Fecha: .....

# Agradecimientos

Este trabajo de fin de grado es el fruto de muchas horas de dedicación el cual no habría sido posible sin la ayuda de muchas personas.

En primer lugar agradecer los videos proporcionados por el grupo de investigación GEINTRA los cuales han sido cruciales para el desarrollo de este trabajo.

Destacar toda la ayuda proporcionada por las profesoras Cristina Losada y Marta Marrón que siempre han estado disponibles para cualquier duda que me surgiese.

También quiero agradecer a Marcos Baptista todo lo pendiente que ha estado de mi ante cualquier duda que tenía y a todos mis compañeros de la universidad por tantos cafés que han hecho que desconectase y me relajase.

Y por último, decir gracias a toda mi familia por la paciencia enorme que han tenido conmigo. A mi padre, por todos los consejos que me ha dado, a mi madre por escuchar todas mis explicaciones aunque le “sonasen a chino”, a mi hermano por saber hacerme reir y desconectar y a Leti, por aguantarme más de lo que debería y hacer que viese que todo es más fácil de lo que creía. Este trabajo nos pertenece a los cinco.

# Resumen

El objetivo de este trabajo de fin de grado es la implementación de un demostrador para la detección de personas y el análisis de su actividad en imágenes de color. Dicha implementación se divide en dos módulos.

En el primero de ellos se diseña un sistema de detección y seguimiento de personas usando descriptores HOG, un clasificador SVM y un banco de filtros de Kalman. A continuación, por cada persona detectada se extraen secuencias que se emplean como entrada al segundo módulo, el cual implementa un clasificador para la detección de distintas actividades realizadas por humanos.

Para evaluar el sistema completo se han realizado múltiples pruebas experimentales utilizando secuencias de vídeo realistas.

**Palabras clave:** Detección y seguimiento, reconocimiento de la actividad humana, visión artificial, descriptores HOG, clasificadores SVM, filtro de Kalman, Action Bank, OpenCV.

# Abstract

The main aim of this work is the implementation of a demostrator for people detection, tracking and activity recognition. The developed demostrator is divided into two different modules.

The first module includes the people detection and tracking stages based on HOG features, a SVM classifier and a bank of Kalman filters. For each detected person, a video sequence is extracted in order to use it as input for the people recognition module.

In order to evaluate the developed system, several experimental tests have been carried out using a dataset of realistic video sequences.

# Índice general

# Índice de figuras

# Índice de tablas

# Índice de algoritmos

# Capítulo 1

# Introducción

## 1.1 Contexto

El estado actual de las tecnologías de procesamiento de audio y vídeo permite abordar tareas cada vez más complejas y con mayores niveles de precisión. La comunidad científica internacional realiza importantes esfuerzos para ir más allá de la simple extracción de información acústica o visual, para obtener información semántica, relacionada con los eventos que están teniendo lugar en un espacio determinado, en el contexto de lo que se ha dado en llamar interpretación de escenas (*scene understanding*), con especial énfasis en las tareas centradas en el reconocimiento de la actividad humana.

El reconocimiento de las actividades realizadas por personas es actualmente uno de los objetivos principales de la visión artificial y está motivado por el potencial de sus muchas aplicaciones tales como la identificación de personas, el control de aforos o la detección de eventos anómalos. Los sensores más utilizados son cámaras de color y profundidad y las características que se extraen de las secuencias obtenidas para realizar un reconocimiento son tanto de bajo como de alto nivel.

En este contexto, el presente trabajo de fin de grado (TFG) se enmarca dentro de las líneas del grupo de investigación de Ingeniería Electrónica Aplicada a Espacios Inteligentes y Transporte (GEINTRA) [6], y tiene como objetivo el desarrollo y evaluación de un demostrador para la detección y seguimiento de personas y el reconocimiento de las acciones realizadas por esas personas a partir de secuencias de imágenes de color. Así, se ha implementado un sistema que, a partir de una secuencia de imágenes RGB permite la detección y seguimiento de todas las personas que aparezcan en dicha secuencia. Posteriormente, por cada una de las personas se realiza el análisis de segmentos de vídeo para, mediante el uso de un clasificador, determinar la actividad que está realizando.

Cabe destacar que el trabajo aquí descrito, toma como punto de partida y continua la línea de investigación iniciada con los trabajos previos realizados dentro del grupo GEINTRA por Marcos Baptista Ríos [4] para la etapa de detección y seguimiento, y Carlos Martínez García [8] para el módulo de reconocimiento de acciones.

## 1.2 Objetivos y solución propuesta

En el contexto descrito, el objetivo de este trabajo es la creación de un demostrador en el que a partir de un conjunto de imágenes sea posible determinar la acción que está realizando cada persona presente en la secuencia, mediante un banco de acciones, y mostrar los resultados de forma gráfica sobre la misma.

En la figura 1.1 se representa de manera resumida los módulos de los que consta el sistema que se ha empleado para el demostrador desarrollado. En él se observa que dada una secuencia de imagenes, el primer paso es la detección y seguimiento de las personas que aparecen en la misma. Para la detección se emplean descriptores basadas en histogramas de gradientes orientados (HOG) [1, 9] y un clasificador basado en una máquina de soporte vectorial (SVM) [2]). El seguimiento de dichas personas se realiza mediante un banco de filtros de Kalman. Por cada una de las personas detectadas, se debe extraer una región de interés (ROI) espacio-temporal, que se emplea para posteriormente extraer la acción que se está realizando con la ayuda del banco de acciones ActionBank [3, 7]. El último paso es la reconstrucción del vídeo original a partir de los segmentos espacio-temporales del mismo, y la representación gráfica de los resultados.

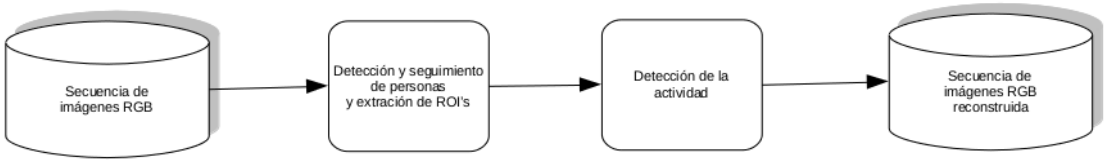



Figura 1.1: Esquema resumen de los módulos del proyecto

Para las tareas de detección y seguimiento de personas [4] así como del reconocimiento de acciones [8] se hará uso de trabajos previos desarrollados en el grupo de investigación GEINTRA [6] (Grupo de Ingeniería Electrónica Aplicada a Espacios Inteligentes y Transporte).

## 1.3 Organización de la memoria

La presente memoria describe los aspectos teóricos y prácticos necesarios para alcanzar el objetivo propuesto, así como los principales resultados y conclusiones. Para ello este escrito se divide en los siguientes apartados:

- **Capítulo 1 - Introducción:** en este apartado se realiza una introducción para contextualizar el trabajo de fin de grado (TFG) dentro del ámbito de estudio así como la solución propuesta.
- **Capítulo 2 - Fundamentos Teóricos:** recoge de manera teórica y matemática los distintos módulos que intervienen en el demostrador, realizando un estudio de las distintas técnicas de detección de personas (Sección 2.1) y su seguimiento (Sección 2.2) así como un estudio de los distintas técnicas existentes para la clasificación de datos (Sección 2.1.2).
- **Capítulo 3 - Desarrollo:** en este capítulo se describe el procedimiento llevado a cabo en el desarrollo de este trabajo de fin de grado así como las justificaciones relevantes en cuanto a las decisiones tomadas a la hora de idear las pruebas realizadas.
- **Capítulo 4 - Resultados:** en el que se exponen los resultados obtenidos en las diversas pruebas experimentales realizadas.
- **Capítulo 5 - Conclusiones y líneas futuras:** recoge las principales conclusiones a las que se han llegado tras la realización de este trabajo así como algunas posibles líneas de trabajo futuro.

# Capítulo 2

# Fundamentos teóricos

*Y así, del mucho leer y del poco dormir, se le secó el cerebro de manera que vino a perder el juicio.*

Miguel de Cervantes Saavedra

En este capítulo se pretende exponer todos los fundamentos teóricos que envuelven el desarrollo del trabajo de fin de grado propuesto en esta memoria. Para ello se ha profundizado en los campos de la detección de personas y su seguimiento, así como para el reconocimiento de actividades humanas. En ambos casos, se explican tanto los descriptores elegidos, como el clasificador utilizado.

## 2.1 Detección de personas

Un sistema de detección de personas visual se encarga de determinar si en una imagen existe o no la presencia de una persona. En caso afirmativo de debe de proporcionar tanto la posición como el tamaño de dicho individuo dentro de la imagen. La estructura que sigue dicho bloque es la que se muestra en la figura 2.1 y se compone de un módulo de extracción de información de la imagen y otro de clasificación. cada uno de estos módulos se describe con mayor detalle en los siguientes apartados.

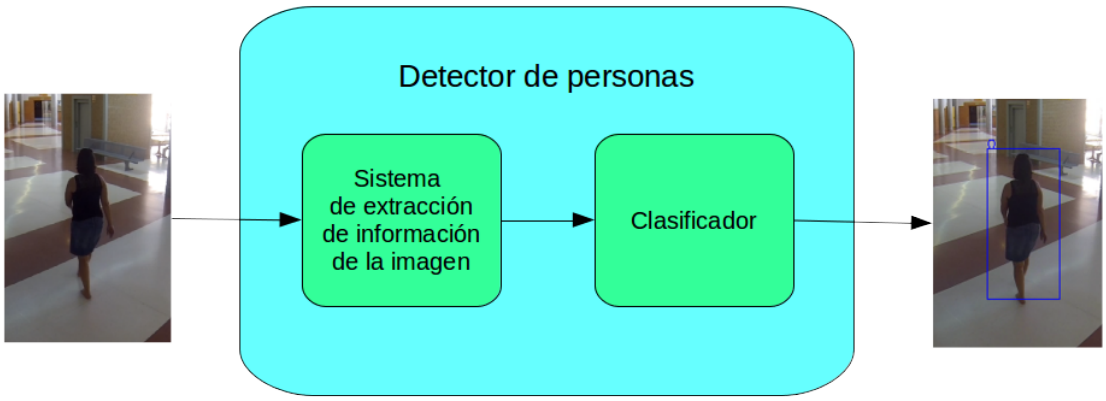



Figura 2.1: Estructura del seguidor de personas

### 2.1.1 Extracción de descriptores de una imagen

En el análisis de imágenes es crucial el poder traducir la imagen de entrada en información o descriptores que puedan ser tratados por el sistema y que represente fielmente la realidad. Un descriptor se trata de un conjunto de valores numéricos o de atributos que se asocian a la imagen o elemento segmentado de entrada. Existen dos grandes grupos de descriptores:

- **Descriptores de información general:** consisten en descriptores que cubren distintas características visuales básicas y elementales como son el color, la textura, la forma, el movimiento y la localización entre otras.
- **Descriptores de información de dominio especifico:** estos descriptores, que proporcionan información sobre objetos y eventos en la escena, no son fáciles de extraer, aún más si se pretende realizar una extracción automática y para ello se utilizan distintas técnicas de extracción.

Según los distintos tipos de técnicas empleadas para extraer los descriptores de una imagen se pueden encontrar los diferentes tipos de detectores para realizar la tarea posterior de detectar personas:

- **Detectores basados en segmentación:** hacen uso de un conocimiento a priori del fondo para separarlo del primer plano que correspondería a las personas a detectar. Este método presenta múltiples inconvenientes y es muy poco robusto a cambios de iluminación, fondos dinámicos y movimientos de la cámara.
- **Detectores basados en características de forma:** recurren al modelo de forma implícita o ISM (*Implicit Shape Model*) para obtener las localizaciones de las personas en base a votaciones de un conjunto de características similares a las de un diccionario previamente aprendido. Este método sólo funciona de manera adecuada para imágenes de alta resolución, obteniendo resultados muy pobres cuando la calidad de la imagen no es demasiado alta.
- **Detectores basados en ventana deslizante:** proporcionan un mejor rendimiento para baja y media resolución. Esta técnica consiste básicamente en recorrer una imagen seleccionando cada vez una determinada región local, a la que se denominará ventana, y para la cual se extraerá un descriptor de características que nos permitirá saber si dicha región delimita el cuerpo de una persona o no. Para conocer si una ventana contiene una persona, se recurrirá al uso de un clasificador previamente entrenado.

En este proyecto se utilizará la técnica de ventana deslizante mediante con una obtención de características basadas en histogramas de gradientes orientados (HOG) [9], ya que aunque es muy básica, proporciona unos resultados satisfactorios además de ser actualmente una técnica referente en los sistemas de detección.

El descriptor HOG se basa en la idea de que la apariencia local de una imagen puede ser caracterizada de manera adecuada por una distribución de gradientes o direcciones de las aristas a nivel local, incluso con un conocimiento pobre de su posición. Tomando esto como partida, se desarrolla un tipo de descriptor consistente en histogramas locales de orientación de los gradientes de regiones solapadas y distribuidas de una imagen, o de una región de la misma, a la cual se denomina ventana de detección. El proceso de extracción de los descriptores HOG se basa en evaluar un conjunto normalizado de histogramas locales de orientación de los gradientes (figura 2.2).

El proceso de extracción del descriptor HOG consta de 3 fases:

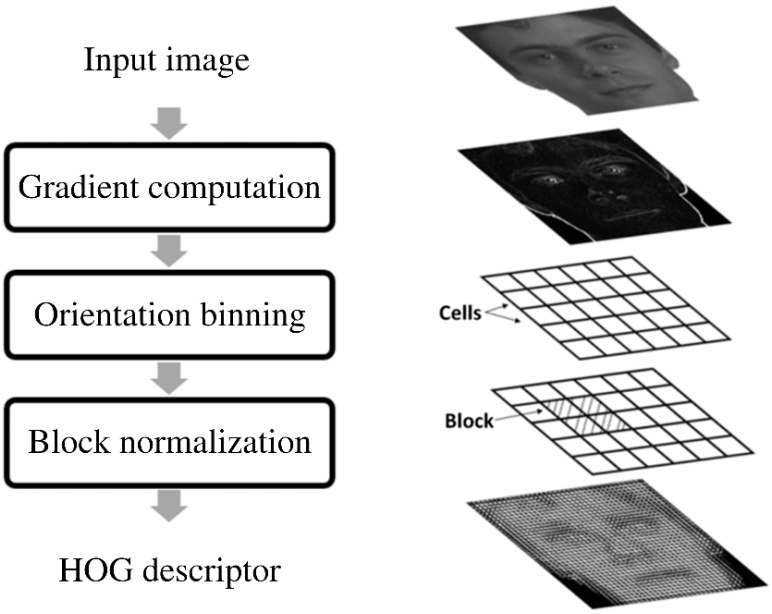



Figura 2.2: Proceso de extracción de descriptores HOG [1]

1. **Cálculo del gradiente.**
   El primer paso para el cálculo de descriptores es el de normalizar los valores de color y gamma. Tras esto se trata de capturar la información del contorno y silueta de la imagen mediante el calculo de gradientes de primer y segundo orden. Para imagenes de color RGB se hace este proceso por cada canal siendo el resultado el gradiente del canal que mayor norma presente. Lo más habitual es que este cálculo se realice aplicando una máscara derivativa de primer orden.

2. **Codificación de los histogramas locales de orientacion.**
   El segundo paso es la creación de los histogramas locales de cada celda. La codificación utilizada para las características es localmente sensible al contenido de la imagen. Para cada celda se acumula el histograma de la orientación de los gradientes contenidos en ella. La orientación que sea mayoritaria se considerará como la orientación de la celda.

3. **Normalización del bloque.**
   La etapa de normalización del bloque permite mejorar la invarianza a la iluminación y al sombreado, así como el contraste de los bordes. Los distintos esquemas de normalización que se suelen utilizar son la norma L2 (ecuación 2.1) y la norma L1 (ecuación 2.2).

$$v_{L2} = \frac{v}{\sqrt{||v||_2^2 + e^2}} \tag{2.1}$$

$$v_{L1} = \frac{v}{||v||_1 + e} \tag{2.2}$$

### 2.1.2 Clasificación de imágenes

Un clasificador se utiliza para, dada una serie de características de un subconjunto (en nuestro caso un vector de descriptores HOG), ser capaz de clasificar en distintas clases el resto del conjunto. La clasificación puede ser binaria (si solo existen dos clases) o multiclase. En el caso de la detección de personas solo se detectará la existencia o no de personas, por lo que se tratará de un clasificador binario. Dentro de los clasificadores existen diferentes tipos:

- **Clasificadores supervisados:** técnica de aprendizaje artificial que elabora una función matemática (hipótesis) a partir de datos de entrenamiento previamente etiquetados. Donde el usuario con unos datos de entrenamiento en una máquina puede deducir entre un conjunto de datos de entrada a que clase pertenecen los datos de salida.
- **Clasificadores no supervisados:** no disponen de un conjunto de entrenamiento que permita conocer las etiquetas de los datos, así pues, se hace necesario el uso de técnicas de agrupamiento que intentan construir estas etiquetas. Este sistema de agrupamiento (o *clustering*) tiene como finalidad catalogar los objetos en conjuntos tales que los que estén en el mismo sean muy semejantes entre sí, mientras que el grado de semejanza entre grupos diferentes sea bajo. Aun así, uno de los problemas que presenta este método es la toma de decisiones a la hora de escoger un patrón entre todos los proporcionados.
- **Clasificadores semi-supervisados:** surge como consecuencia de la dificultad que se tiene para obtener los datos etiquetados requeridos por los métodos supervisados, ya que éstos deben ser etiquetados por un experto de forma manual convirtiéndose en un trabajo pesado. Este aprendizaje es una combinación del aprendizaje supervisado y no supervisado. Puesto que asignar etiquetas o clases de datos puede ser muy costoso, el aprendizaje semi-supervisado recurre a la opción de usar a la vez una colección limitada de datos etiquetados y un conjunto más extenso de datos no etiquetados, mejorando así la construcción de los modelos. En este método, se ha de tener en cuenta que no siempre los datos no etiquetados son de ayuda al proceso de aprendizaje. Por lo general se asume que los datos no etiquetados siguen la misma distribución que los etiquetados para que el uso de datos sin etiquetar sea útil.
- **Clasificadores de refuerzo:** En este tipo de aprendizaje, el agente dispone de un conjunto de opciones disponibles para llegar a resolver una determinada tarea. Para ello, debe escoger en primer lugar, la opción que considere más oportuna para lograr su objetivo. Una vez hecho, el agente en cuestión recibe información de retroalimentación del entorno sobre su desempeño. Posteriormente, dicha información puede ser utilizada para modificar su comportamiento. El aprendizaje por refuerzo conlleva pues, una fuerte carga de ensayo y error.

El clasificador que se utilizará en este trabajo de fin de grado con el proposito de detectar a personas es el conocido como máquina de soporte vectorial o SVM. Este clasificador pertenece a los modelos de aprendizaje supervisado y trata de construir un hiperplano o un conjunto de hiperplanos en un espacio de dimensión superior para conseguir separar un conjunto de datos en distintas clases según sus características. Para que la clasificación sea óptima estos hiperplanos deberán buscar la mayor distancia de separación entre las muestras que dividen para conseguir así minimizar errores. El caso más básico es cuando se trata de un clasificador SVM binario, es decir, sólo se quieren clasificar dos clases distintas. En este caso, existen infinitos planos que dividan las muestras en dos clases, sin embargo la solución óptima será la que consiga dividirlas con un margen máximo, esto puede observarse en la figura 2.3. En ella se puede observar como el hiperplano H1 no consigue separar correctamente las clases, H2 consigue separarlas pero con un margen pequeño y H3 las separa con el margen máximo.

Por lo tanto el problema de optimización lineal, puede definirse como que a partir de un conjunto de muestras de entrada $x \in \mathbb{R}^n$, pertenecientes a las diferentes clases $y \in \mathbb{N}$, se desea encontrar el hiperplano de parámetros $\{w, b\}$, que realice la mayor separación entre las clases. Esto se pone de manifiesto en la siguiente ecuación y representado en la figura 2.4.

$$wx_i + b \gtrless y_i \;\; / \;\; \begin{cases} y_i = -1 & \forall x_i \in C_i = B \\ y_i = 1 & \forall x_i \in C_i = A \end{cases} \Rightarrow (w^*, b^*) = 0 \rightarrow H_0 \tag{2.3}$$

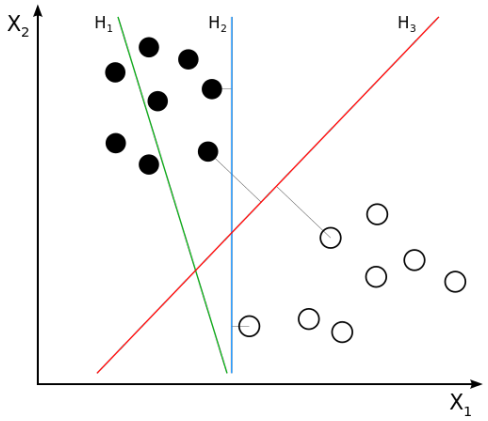



Figura 2.3: Ejemplo de distintos hiperplano para clasificación SVM linear [2]

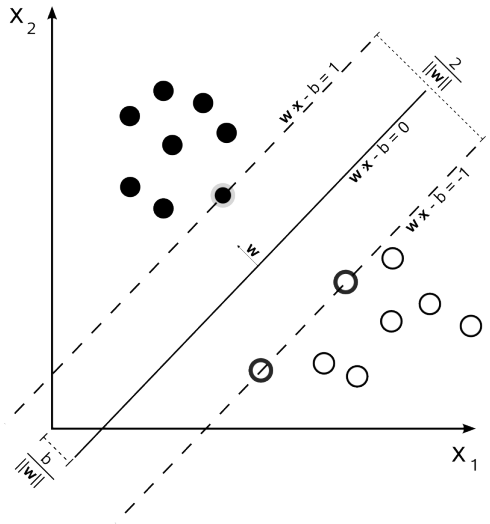



Figura 2.4: Obtención del hiperplano y márgenes [2]

La separación perfecta del hiperplano en dos clases bien diferenciadas y con un margen máximo no siempre es posible o si lo es no puede ser generalizado a otros datos (*overfitting*). Con el fin de permitir cierta flexibilidad, los SVM manejan un parámetro ($C$) que controla la compensación entre errores de entrenamiento y los márgenes rígidos, creando así un margen blando (*soft margin*) que permita algunos errores en la clasificación a la vez que los penaliza.

Aunque para el desarrollo matemático solo se emplean dos clases $A, B$, es posible realizar una clasificación multiclase empleando una de las siguientes estrategias:

- ***One Against All*:** se diseña una SVM binaria por cada una de las clases y se entrena para diferenciar entre la existencia de cada una de esas clases respecto al resto, de tal manera que la salida se decide por maximización de las salidas.
- ***One Against One*:** se diseñan $\frac{1}{2}C(C-1)$ clasificadores binarios. Cada SVM binaria distingue entre cada par de clases y se decide el resultado mediante votación. En caso de que no se llegue a decidir una salida por producirse un empate, se recurre a la estrategia *One Against All* para decidir entre las clases empatadas.

En muchos casos prácticos, los algoritmos SVM no trabajan con solo dos clases ni pueden aplicar los métodos descritos anteriormente para clasificación multiclase. Esto se hace presente cuando:

1. Existen más de dos variables predictoras.
2. El universo de estudio no es linealmente separable.
3. El conjunto de datos presentado no es separable.

La representación por medio de las funciones Kernel ofrece una solución a este problema, proyectando la información a un espacio de características de mayor dimensión, es decir, se mapea el espacio de entradas $X$ a un nuevo espacio de características de mayor dimensionalidad.

Algunos de las funciones Kernel mas utilizadas son las mostradas en la tabla 2.1.

| **Tipo** | **Función Kernel** |
|---|---|
| Polinómica | $(x^T y + 1)^p$ |
| RBF gausianas | $e^{-(\frac{\lvert x-x_j \rvert^2}{2\sigma^2})}$ |
| RBF exponenciales | $e^{-(\frac{\lvert x-x_j \rvert}{2\sigma^2})}$ |

Tabla 2.1: Tipos de funciones Kernel [4].

## 2.2 Seguimiento de personas

A pesar de la gran variedad de detectores existentes, en el caso de los vídeos, el análisis individual de las imágenes no produce resultados del todo satisfactorios. Por ello se hace necesario un sistema de seguimiento que mejore la calidad de las detecciones obtenidas y a su vez asocie dicha detección con otras pasadas. Los sistemas que son capaces de llevar acabo estos requisitos se les conoce como sistemas de seguimiento mediante detección.

De una manera básica se puede considerar como un sistema de seguimiento aquel que obtiene una estimación de la trayectoria de un cuerpo según se desplaza por la escena. Los métodos mas utilizados para dar solución al problema son los métodos probabilísticos de estimación o estimadores. Algunos de los más utilizados son:

- Mínimos cuadrados recursivos (LMS)
- Filtro de Kalman (KF)
- Filtro de partículas (PF)

En este proyecto de fin de grado se emplea un estimador basado en el filtro de Kalman desarrollado en [4].

A pesar de que el rendimiento del filtro de Kalman es óptimo en cuanto al seguimiento de objetos, es cierto que las trayectorias que se consiguen tienen transiciones rápidas que pueden ser confusas a la hora del entrenamiento del clasificador de actividad posteriormente descrito en la sección 2.3, por lo tanto es conveniente suavizar la forma de la trayectoria mediante algún tipo de filtro.

El tipo de filtro que realiza esta función es un filtro paso bajo que atenúe dichas transiciones no deseadas, por ello se considera como el filtro ideal para esta aplicación el denominado filtro de promediado móvil ponderado exponencialmente o EWMA [10] [11] (Exponentially Weighted Moving Average) debido a su fácil implementación y su reducido coste computacional.

### 2.2.1 Filtro de Kalman

El objetivo principal del filtro de Kalman es el de obtener la estimación de un vector de estado variante en el tiempo a partir de observaciones con incertidumbre de dicho vector. Este filtro se caracteriza por las siguientes ecuaciones:

$$\hat{\mathbf{x}}[n+1] = \mathbf{A}\mathbf{x}[n] + \mathbf{B}\mathbf{u}[n+1] + \mathbf{w}[n] \tag{2.4}$$

$$\hat{\mathbf{z}}[n+1] = \mathbf{H}\mathbf{x}[n+1] + \mathbf{v}[n+1] \tag{2.5}$$

De las ecuaciones 2.4 y 2.5 se sabe que:

- **u**[n] es el vector de entrada.
- **x**[n] es el vector de estado.
- **z**[n] es el vector de salida.
- **w**[n] y **v**[n] representan el ruido del proceso y el ruido de la medida respectivamente. Ambos son ruidos blancos gaussianos, de modo que permiten al filtro alcanzar una solución estable.
- **A**(r x r): Es la matriz de actualización de estado en función del estado anterior.
- **B**(r x l): Es la matriz de actualización de estado en función de la entrada.
- **H**(s x r): Es la matriz que relaciona el estado con la medida.

Para todos los casos, $r$ tendrá el valor de la longitud del vector de estado $x$[n], $l$ tendrá el valor del vector de entrada $u$[n] y $s$ tendrá el valor de la longitud del vector de salida $z$[n]. Para el caso básico las matrices **A**, **B** y **H** se consideran constantes. El proceso de estimación de este filtro se basa en un algoritmo recurisvo que consta de dos etapas:

- **Predicción:** Las ecuaciones de esta etapa adelantan el estado actual y el error de covarianza estimados para obtener la estimación a priori en el siguiente paso
- **Corrección:** Las ecuaciones en este caso utilizan las estimaciones obtenidas en la etapa de predicción y con una nueva medida realizada se mejora dicha estimación a posteriori.

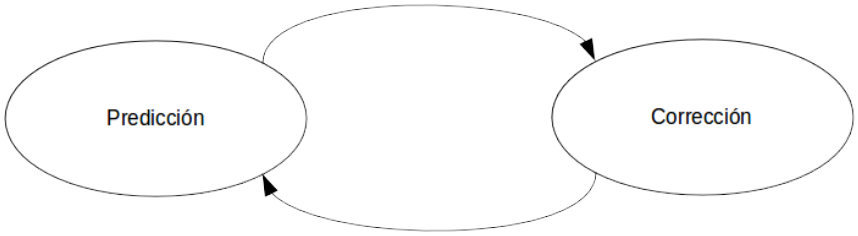



Figura 2.5: Proceso Iterativo de estimación del filtro de Kalman

Para los casos en los que el objeto a detectar se desplaza a una velocidad constante se puede asumir el modelo de velocidad constante en el que se incluye dentro del vector de estados la velocidad del objeto bajo estudio. Para los casos en la que la velocidad no sea constante del todo (la mayoría de ellos) se consideran esas fluctuaciones de velocidad como ruido, pudiendo así seguir con la misma hipótesis.

### 2.2.2 Asociación de identidades

La asociación de las distintas detecciones en identidades distintas que mantenga la coherencia del seguimiento es muy importante, ya que una asociación incorrecta podría propiciar que las estimaciones posteriores sean erróneas, por lo que el algoritmo que deberá buscar una solución óptima entre todas las posibles.

Los algoritmos de asociación más utilizados son los que se describen a continuación:

- ***Multiple Hypotheses Tracker* (MHT)**: este algoritmo de *tracking* (o seguimiento) se basa en construir un árbol de hipótesis de potenciales trayectorias para cada candidato. Se calcula la probabilidad de cada trayecto y se selecciona la trayectoria completa como la combinación de los trayectos más probables. Este algoritmo es muy costoso computacionalmente hablando para aplicaciones de visión artificial y principalmente se utiliza en aplicaciones de *tracking* con radar.
- **k *Nearest Neighbours* (k-NN)**: este algoritmo se basa en dado una nube de puntos, se calcula la trayectoria respecto a una estimada obteniendo la distancia euclídea de los puntos y siendo el resultado el de menor distancia.
- ***Probabilistic Data Association* (PDA)**: consiste en un algoritmo probabilístico que obtiene la probabilidad de que una hipótesis esté relacionada con alguna de las estimaciones disponibles.

### 2.2.3 Acondicionamiento de señales en tiempo real: Filtro EWMA

Este filtro de respuesta al impulso infinita recursivo consiste en serie de pesos que decrecen exponencialmente. con esto se consigue que el peso de cada dato a lo largo del tiempo tenga un peso que nunca llega a cero. El filtro debe ser inicializado correctamente, y para ello existen varias estrategias. Una de ellas es forzar la primera salida a tener el valor de la entrada. Con esto se consigue que solo se produzca el retardo de cómputo, otra sería el realizar la media de las cuatro o cinco primeras muestras retardando así el comienzo de funcionamiento del sistema.

El valor de inicialización del filtro puede ser más o menos relevante en función del valor de $\lambda$, siendo este variable entre 0 y 1. A menor valor de $\lambda$ menos sensible es el sistema a las variaciones pronunciadas de los datos de entrada, pero se produce un mayor filtrado de la señal perdiendo su forma de onda, por otro lado cuando el parámetro descrito es alto, se tiene más en cuenta el dato de entrada y perdiendose así eficacia en el filtrado. Los valores típicos para $\lambda$ son 0.05, 0.15, 0.2, 0.25 ó 0.3 dependiendo de la finalidad.

La ecuación que pone de manifiesto dicho filtrado es la que se muestra a continuación:

$$\hat{Y}[n] = \begin{cases} Y[n] & n = 0 \\ \lambda Y[n] + (1-\lambda)\hat{Y}[n-1] & n \geq 1 \end{cases} \Rightarrow 0 \leq \lambda \leq 1 \tag{2.6}$$

En ella se observa como el valor inicial del filtro corresponde con el primer dato de entrada y como el filtro es recursivo, ya que la salida actual depende de la salida anterior. Al desarrollar esta fórmula más en detalle se observa lo siguiente. Suponiendo que $Y[n]$ / $n \in [0\ k]$ se tiene que...

$\hat{Y}[0] = Y[0]$
$\hat{Y}[1] = \lambda Y[1] + (1-\lambda)\hat{Y}[0] = \lambda Y[1] + (1-\lambda)Y[0]$
$\hat{Y}[2] = \lambda Y[2] + (1-\lambda)\hat{Y}[1] = \lambda Y[2] + (1-\lambda)\lambda Y[1] + (1-\lambda)^2 Y[0]$
$\vdots$

$\hat{Y}[k] = \lambda Y[k] + (1-\lambda)\hat{Y}[k-1] = \lambda Y[k] + (1-\lambda)\lambda Y[k-1] + (1-\lambda)^2 \lambda Y[k-2] + \cdots + (1-\lambda)^{(k-1)} \lambda Y[1] + (1-\lambda)^k Y[0]$

De este desarrollo se puede deducir una expresión general de $\hat{Y}[k]$ mostrada en la siguiente ecuación:

$$\hat{Y}[k] = \lambda \sum_{n=0}^{k-1} ((1-\lambda)^n Y[k-n]) + (1-\lambda)^k Y[0] \Rightarrow 0 \leq \lambda \leq 1 \tag{2.7}$$

En ella se puede observar como en función de la muestra esta tiene un peso de $\lambda(1-\lambda)^n$ para $n > 0$. Por otro lado si volvemos a la ecuación 2.6 y obtenemos la respuesta al impulso del filtro:

$$\begin{aligned} \hat{Y}[n] &= \lambda Y[n] + (1-\lambda)\hat{Y}[n-1] \\ \hat{Y}[n] - (1-\lambda)\hat{Y}[n-1] &= \lambda Y[n] \end{aligned} \tag{2.8}$$

Aplicado la transformada $\mathbb{Z}$ a la ecuación anterior se obtiene:

$$\begin{aligned} \hat{Y}(z) - (1-\lambda)z^{-1}\hat{Y}(z) &= \lambda Y(z) \\ \hat{Y}(z)(1-(1-\lambda)z^{-1}) &= \lambda Y(z) \end{aligned} \tag{2.9}$$

Por lo tanto la respuesta al impulso viene dada por:

$$H(z) = \frac{\lambda}{1-(1-\lambda)z^{-1}} \tag{2.10}$$

Finalmente, realizando la transformada $\mathbb{Z}$ inversa se obtiene la expresión del filtro (ecuación 2.11), que coincide con los pesos anteriormente descritos y que se puede observar en la figura 2.6.

$$h[n] = \lambda(1-\lambda)^n \tag{2.11}$$

Por otro lado, en la figura 2.7 se puede observar la calidad del filtro para diferentes valores de $\lambda$, que atiende a la función $\hat{Y}[n] = Y[n] * h[n]$.

Como se puede observar a un menor valor de $\lambda$ la muestra actual tiene un menor peso en y prevalece la influencia de las muestras pasadas, sin embargo la señal original queda muy filtrada perdiendo la forma de onda de los datos de entrada. Por otro lado, a mayor $\lambda$ el dato actual pasa tener mayor relevancia conservándose así mejor la forma de onda.

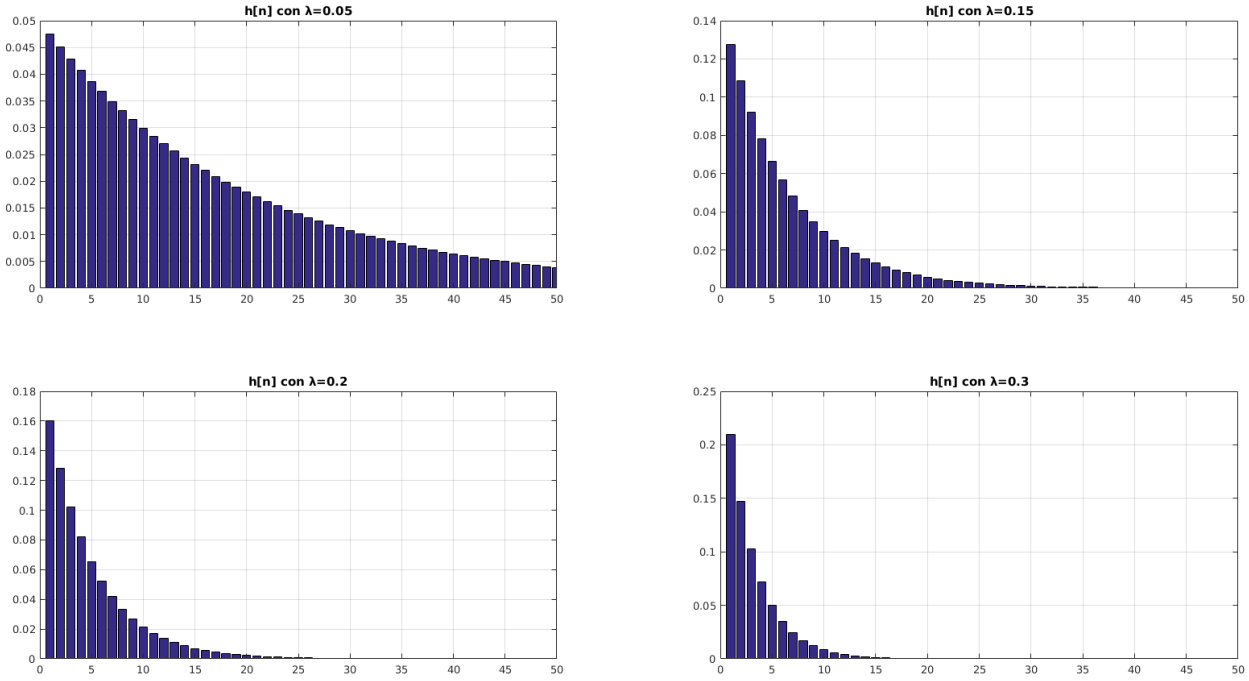



Figura 2.6: Respuesta al impulso para $\lambda = 0.05$, $\lambda = 0.15$, $\lambda = 0.2$ y $\lambda = 0.3$

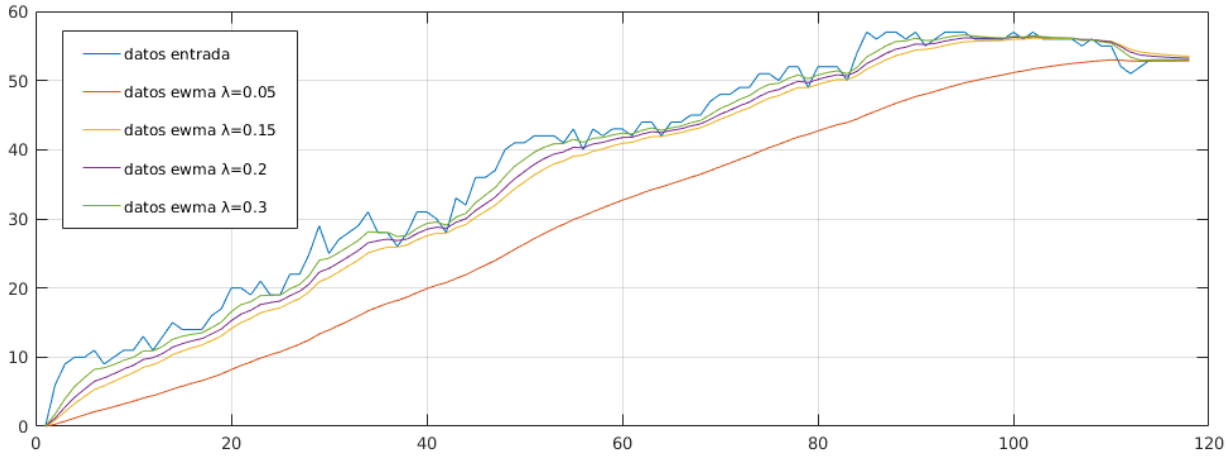



Figura 2.7: Ejemplo de filtro EWMA

## 2.3 Reconocimiento de actividades mediante *ActionBank*

La clasificación de la actividad humana es extremadamente compleja. Para que sea lo más exitosa posible hay que seleccionar adecuadamente las características (*features*) a analizar tales como la localización espacio-temporal, trayectorias o histograma de gradientes en 3D. Estos casos son características de bajo y medio nivel y obtienen rendimientos muy bajos debidos a la cantidad limitada de movimientos que pueden distinguir. Por otro lado también se pueden obtener características de alto nivel para mejorar el rendimiento.

Action Bank$^{TM}$ [3,7] es un método de representación de alto nivel de actividad basado en los bancos de objetos (*Object Bank Method*), las acciones son incluidas en un conjunto de detectores de acciones individuales con varias escalas y puntos de vista. La representación del banco de acciones es una conca-

tenación de los *features* de la cantidad las detecciones agrupadas volumétricamente de cada detector de acción. La acción realizada será la del detector de acción que proporcione un mayor nivel.

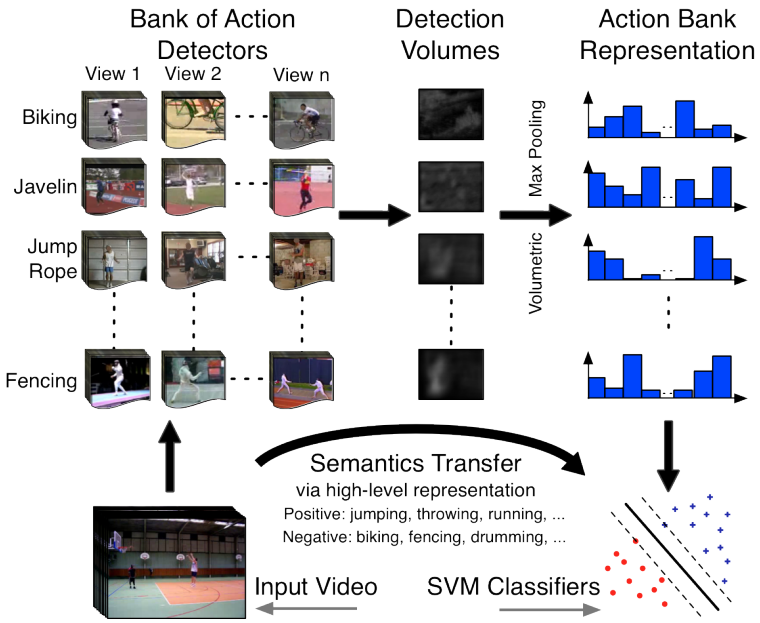



Figura 2.8: Representación del funcionamiento de ActionBank [3].

Según [3] y tal y como muestra la figura 2.9, la salida que proporciona esta herramienta son los volúmenes de energías de los vídeos de entrada. Para obtenerlos, ActionBank$^{TM}$ descompone el vídeo de entrada en siete energías espacio-temporales canónicas: hacia izquierda, hacia derecha, hacia arriba, hacia abajo, cambios rápidos, estático y falta de estructura orientada. Las dos últimas no están asociadas a al movimiento y son usadas para modular a los otros cinco para mejorar el poder discriminatorio de la representación

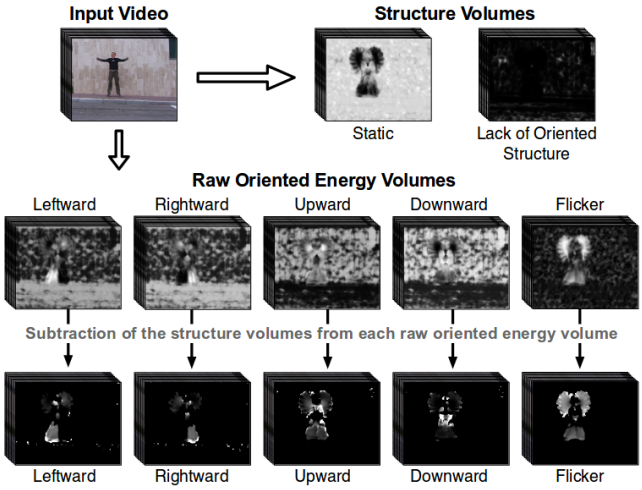



Figura 2.9: Ejemplo de volúmenes de energías calculados por ActionBank [3].

Los descriptores generados por *ActionBank* se clasifican empleando una SVM multiclase entrenada para las diferentes acciones a detectar. Dado que el clasificador es el mismo utilizado para la detección de personas, no se repite aquí el fundamento teórico del mismo, que ya ha sido explicado en el apartado

2.1.2. Tanto el proceso de entrenamiento como el de evaluación de dicha SVM se describen en detalle en el capítulo 3.

# Capítulo 3

# Desarrollo

*A fuerza de construir bien, se llega a buen arquitecto.*

Aristóteles

Como ya se ha comentado en la introducción, el demostrador para la detección de personas y el reconocimiento de acciones en imágenes de color que se propone en este TFG consta de fases distintas para su correcto funcionamiento, tal como se muestra en el esquema general representado en la figura 1.1. En el primero de los módulos se realiza la detección y seguimiento de personas, así como la extracción de regiones de interés (ROIs) que son segmentos de vídeo, en los que debe aparecer una persona realizando una determinada acción. Estas ROIs son las que se emplean en el módulo de detección de acciones. La correcta elección del tamaño (ancho x alto) y duración de estos segmentos de vídeo es fundamental para el funcionamiento del sistema desarrollado.

Por otro lado, el demostrador desarrollado requiere de una etapa de entrenamiento del detector de acciones. Para dicha etapa, también es necesario generar un conjunto de ROIs, siendo, en este caso, imprescindible que los segmentos de vídeo seleccionados contengan una única persona, que realice una única acción acotada. Por este motivo, ambos módulos constan de dos etapas diferentes:

- Una primera etapa de entrenamiento, que se realiza *off-line*, y se encarga de generar las ROIs de los vídeos seleccionados para el entrenamiento (de forma supervisada) para asegurar que cumplen los requisitos impuestos, obtener los descriptores *ActionBank* correspondientes, y entrenar la SVM de la etapa de reconocimiento de acciones.
- Una segunda etapa de test (*on-line*), que se ejecuta de forma automática para un vídeo dado, y lleva a cabo la detección y seguimiento de personas, la extracción de ROI y la detección de acciones utilizando la SVM entrenada en la etapa *off-line*. Esta segunda etapa es la que permite comprobar el correcto funcionamiento del demostrador desarrollado.

En los siguientes apartados se explica el procedimiento llevado a cabo para el desarrollo de cada uno de los módulos.

## 3.1 Detección y seguimiento de personas y extracción de ROI's

En este apartado se explica como se realiza el proceso de detección de personas, el seguimiento de las mismas y la extracción de las regiones de interés (ROIs) deseadas. Como ya se ha comentado, estas ROI's

son utilizadas posteriormente para analizar la acción que se realiza la persona detectada en ellas (ver sección 3.2).

Como ya se ha comentado, la extracción de ROIs se realiza de forma diferente en función de si los segmentos de vídeo extraídos se emplean para el entrenamiento o para la validación del proceso de detección de acciones. La figura 3.1 muestra el esquema general de este módulo (incluyendo las dos etapas mencionadas).

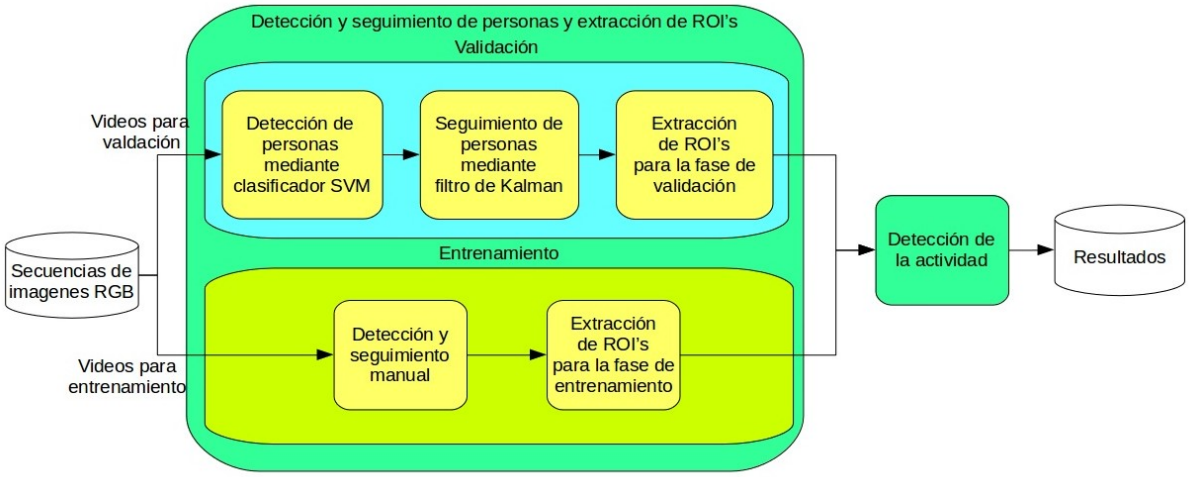



Figura 3.1: Esquema general del funcionamiento del módulo de detección y seguimiento de personas y extracción de ROI's

Aunque los submódulos de entrenamiento y de validación tienen la misma funcionalidad, la de extraer las secuencias en las que aparecen personas realizando alguna acción, se observa en la figura 3.1 que los procesos para llevarlo a cabo son distintos. Esto se debe a que la extracción automática de las secuencias explicadas en la sección 3.1.2 no puede asegurar que la acción realizada esté acotada, pudiendo haber acciones como por ejemplo andar y sentarse o andar y caerse. Este tipo de acciones se pueden observar en la figura 3.2.

Figura 3.2: Ejemplo de acción no acotada.

Por este motivo se ha decidido realizar la fase de detección y seguimiento de manera manual para asegurar que las acciones estén acotadas dentro de una misma secuencia.

### 3.1.1 Extracción de secuencias para la fase de entrenamiento

Tal y como se ha comentado anteriormente, debido a que una extracción automática de las regiones de interés no asegura completamente que se desarrolle una acción acotada en el tiempo que dura la secuencia,

se opta por relizar el proceso de delimitar dichas regiones manualmente. Para ello se utiliza la aplicación de *Matlab* desarrollada en el TFG de Valeria Boggian Arévalo [5], la cual permite realizar el etiquetado de los "bounding box" utilizados para la extracción de las regiones de interés de manera manual. Estos "bounding box" son un conjunto de datos que definen el tamaño y posición de la persona o personas: las coordenadas de la esquina superior izquierda, el alto y el ancho, todas expresadas en píxeles tal y como se muestra en la figura 3.3.


Figura 3.3: Características del "bounding box" [4]

Para realizar el etiquetado completo de un vídeo de entrada, basta con cargarlo en la aplicación siguiendo las indicaciones de losventanas emergentes que aparecen al comienzo de la ejecución, donde se elige el directorio de trabajo así como el vídeo a etiquetar. Una vez elegido, se procederá a extraer los "*frames*" de la secuencia completa si no están disponibles en el directorio de trabajo con anterioridad y a continuación se procede al etiquetado manual.

Para ello, la aplicación utilizada obtiene los "*bounding box*" mediante la selección manual de la esquina superior izquierda y de la inferior derecha de cada persona que aparece en la imagen, así como la acción realizada. Además es posible llevar a cabo un interpolado de las coordenadas de dichos "*bounding box*" entre "*frames*" no consecutivos para agilizar dicho proceso. La figura 3.4 muestra un ejemplo de escena en la que se marcan distintas acciones manualmente.

Los distintos colores apreciados en los "*bounding box*" de la figura 3.4 son un código el cual ayuda al etiquetado automático realizado posteriormente. La aplicación usada devuelve al finalizar su ejecución un archivo de "*Ground Truth*" que es utilizado como histórico de todas las acciones que aparecen en el vídeo, guardando información de la posición de la esquina superior izquierda y de la inferior derecha del "*bounding box*", así como de la acción realizada y de los "*frames*" en los que se desarrolla, de tal manera que, en resumen, se obtienen las trayectorias en las que se realiza una acción ordenadas en el espacio y en el tiempo.

#### 3.1.1.1 Extracción de ROIs

El proceso de extracción de ROIs es fundamental para el desarrollo del demostrador, ya que permite la ejecución automática de ambos módulos. Este proceso ha sido desarrollado íntegramente en este trabajo de fin de grado y su finalidad es la de extraer las secuencias en las que se han detectado personas a lo largo de una secuencia.

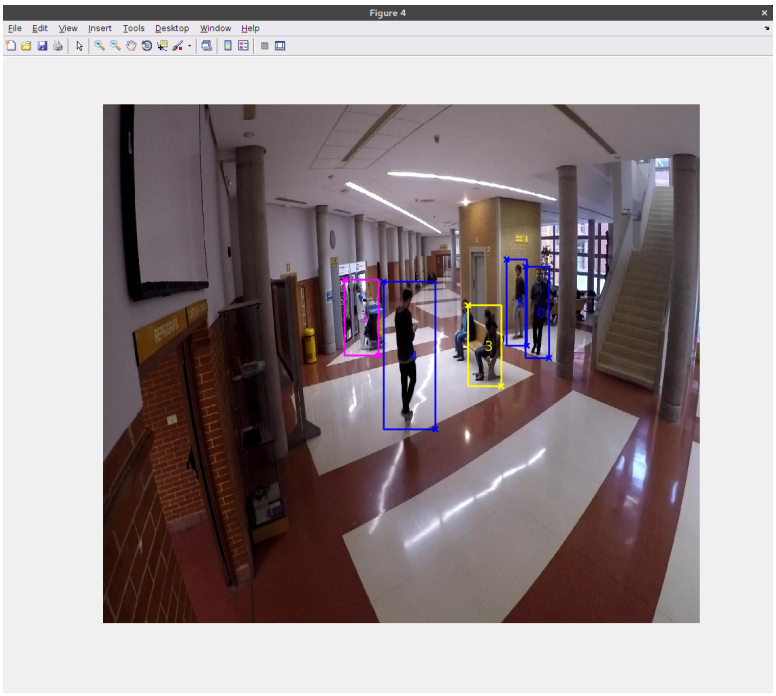



Figura 3.4: Ejemplo de frame etiquetado con la aplicación desarrollada por Valeria Boggian Arévalo [5].

Con la información obtenida del etiquetado manual se procede a extraer las regiones de interés para formar los vídeos que posteriormente se utilizarán para el entrenamiento del clasificador SVM. Para llevar a cabo esto en un primer lugar hay que dividir las trayecciones proporcionadas por el archivo "*Ground Truth*" por duraciones, para ello se sigue la norma que se expone en [12] en la que se recomienda que para el análisis de la acción, los vídeos de acciones no deben tener nunca una duración inferior a un segundo. Al ser la finalidad ultima de estos vídeos la de entrenamiento, se ha decidido extraer secuencias con una duración mínima de un segundo y una máxima de tres. Esto se ha decidido así debido a que la duración de las escenas acotadas es variable, existiendo aciones que duran cerca de un segundo y otras que su duración se extiende más.

Una vez obtenidas las trayectorias y divididas por duraciones, se filtran con el filtro EWMA [10] [11] para evitar así transiciones rápidas en las trayectorias y robustecer el sistema de detección de actividad ante errores por movimientos bruscos de la secuencia de vídeo. La mejora de la trayectoria se pone de manifiesto en la figura 3.5 donde se aprecia como la trayectoria en verde (filtrada) es mucho más suave que la roja (sin filtrar).

Figura 3.5: Ejemplo de suavizado de trayectoria mediante el filtro EWMA

Otro aspecto a tener en cuenta y por lo tanto el último paso a la hora de extraer las ROI's es el de normalizar los tamaños de las regiones de interés obtenidas del archivo "*Ground Truth*". Esto es necesario debido a que los " bounding box" obtenidos en el etiquetado tienen dimensiones distintas tal y como se muestra en la figura 3.6 en la que se observa como se se pueden producir regiones en las que se corta al individuo que realiza la acción.

Figura 3.6: Ejemplo de acción sin la región de interés normalizada.

El proceso de normalización de las regiones de interés se realiza mediante la elección del "bounding box" que sea capaz de contener el mayor area posible de entre todos los "bounding box" existentes dentro de una misma trayectoria, quedando la secuencia tal y como se muestra en la figura 3.7 de tal manera que se evitan dichos cortes en las secuencias.

Figura 3.7: Ejemplo de acción sin la región de interés normalizada.

Una vez realizados todos los pasos anteriormente descritos, las regiones de interés se extraen "*frame*" a "*frame*" de las secuencias delimitadas por dichas ROI's y se crean los videos de acciones con la ayuda de las funciones proporcionadas por *OpenCV*. Una vez obtenidos, gracias al campo de la acción realizada proporcionada por el archivo "*Ground Truth*" se procede a realizar un etiquetado automático de los vídeos, por último para robustecer el clasificador y aumentar el número de muestras, solo para la fase de entrenamiento, se realiza un video en forma especular, otro con la luminosidad aumentada y otro con la luminosidad disminuida tal y como se muestra en la figura 3.8.

### 3.1.2 Extracción de secuencias para la fase de validación

Tal y como se aprecia en la figura 3.1 este submódulo consta de tres procesos que se describen a continuación.

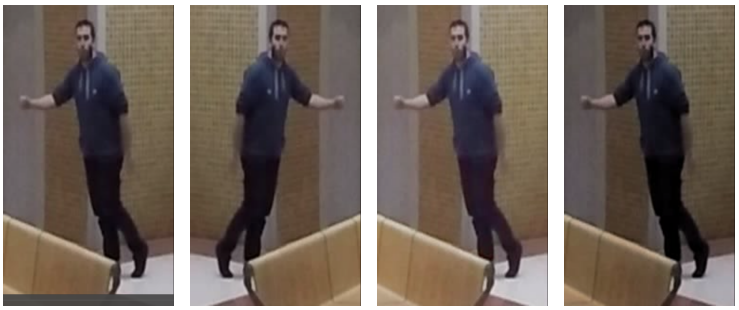


Figura 3.8: Ejemplo de las alteraciones de las secuencias destinadas a entrenamiento.

#### 3.1.2.1 Detección y seguimiento de personas

Para el desarrollo de los dos primeros procesos (detección y seguimiento de personas), se tomó como punto de partida el trabajo fin de grado previo, realizado por Marcos Baptista Ríos [4].

El proceso de detección de personas consta de dos fases, el extractor de características y el clasificador. El primero de ellos hace uso del método de la ventana deslizante con descriptores HOG, descrito en la sección 2.1.1, para extraer las características de la imagen de entrada. Para su implementación se hace uso de las funciones que *OpenCV* [13–15] proporciona para la implementación de dicho método. La implementación en *OpenCV* permite obtener, para cada celda, el módulo y la orientación de los gradientes que posteriormente se ordenan en un histograma de nueve intervalos. El rango total de dicho histograma es de 0º a 180º y normalizado según la norma L2 truncada.

Para realizar la tarea de clasificación también se emplean las funciones proporcionadas por *OpenCV*. En concreto, el clasificador usado está basado en una máquina de soporte vectorial (SVM) lineal de márgen relajado, explicado en la sección 2.1.2, que ha sido previamente entrenado con descriptores HOG. El clasificador proporciona a su salida la distancia al hiperplano de la SVM (que proporciona información acerca de la fiabilidad de la clasificación realizada) además de las características del "bounding box".

En cuanto al proceso de seguimiento se utiliza tanto para robustecer el sistema completo de detección como para tener un histórico de la trayectoria en la escena que realiza el objeto seguido tal y como se explica en la sección 2.2. El seguidor utilizado esta basado en un banco de filtros de Kalman con un modelo de velocidad constante (sección 2.2.1. Debido a que en el escenario de trabajo puede haber detecciones correspondientes a múltiples individuos, para la etapa de seguimiento ha sido necesario realiza una validación y asociación de las medidas proporcionadas por el detector de personas, haciéndolo más robusto ante detecciones erróneas y permitiendo disminuir los errores debidos a oclusiones y cruces de personas.

El resultado de la etapa de seguimiento es un conjunto de trayectorias a lo largo de una secuencia de imágenes, una por cada una de las personas detectadas.

El último proceso es el de la extracción de las ROI's que se realiza a partir de la información del histórico obtenido en la etapa de seguimiento. Este proceso es similar al explicado en la sección 3.1.1.1 con la diferencia de que en este caso las duraciones de las secuencias se podrá elegir paramétricamente, pudiendo elegir entre duración variable (mínima de un segundo y máxima de tres) y duración fija de un segundo. Esto se ha decidido as

## 3.2 Detección de la actividad

Este apartado explica el proceso que se ha seguido para desarrollar el módulo de detección de la actividad que aparece en la figura 1.1. Para realizar la detección y análisis de la actividad se ha decidido partir del trabajo de fin de máster de Carlos Martínez García [8] en el cual se emplean descriptores *ActionBank*, y se implementa un clasificador basado en una máquina de soporte vectorial (SVM). En el trabajo previo [8] se realizaba la fase de entrenamiento y validación de manera conjunta, por cada ejecución del algoritmo. Sin embargo, debido a que en este trabajo se desea que el sistema finalmente desarrollado funcione como un demostrador, ha sido necesario separar dichos módulos tal y como se muestra en la figura 3.9 para facilitar su utilización. Además, también se ha añadido la parte correspondiente a la interpretación y representación de los resultados obtenidos.

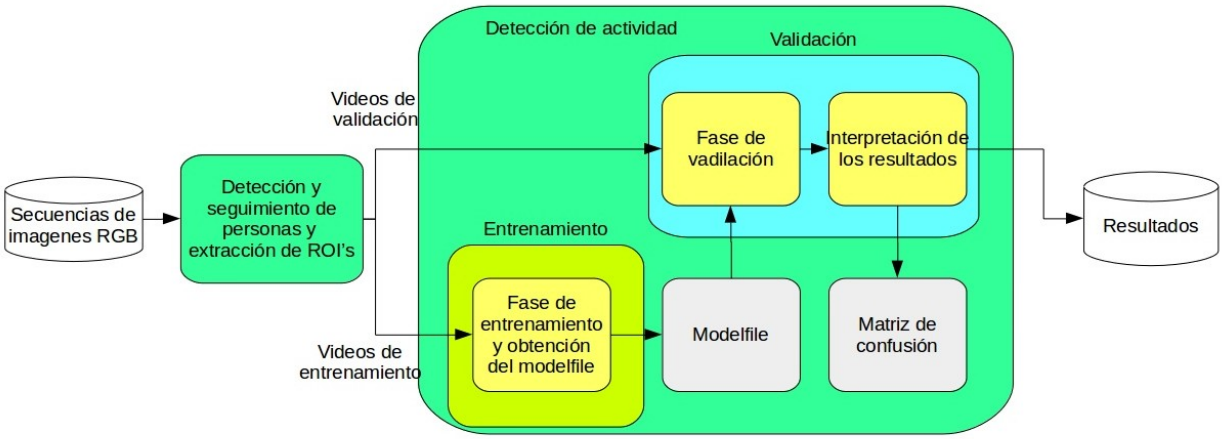



Figura 3.9: Esquema general del funcionamiento del módulo de detección de la actividad

Para este módulo, como ya se ha dicho se utiliza un clasificador SVM, por lo cual hay que extraer de los videos de entradas un conjunto de características para que dicho clasificador funcione correctamente. Estas características son extraidas con ActionBank$^{TM}$ [3, 7] tal como se ha explicado en la sección 2.3.

Este banco de acciones permite escalar las secuencias de imágenes de entrada antes de proceder a la detección de las características. La posibilidad del escalado viene propiciado por el gran tiempo de cómputo asociado a la extracción de estas características, cuanto más pequeña es la imágen de entrada, menos tiempo tarda en ser analizada, pero más error se comete en dicho análisis debido a la pérdida de resolución. Aunque el desarrollador recomienda escalar las imágenes a $\frac{1}{2}$, en este trabajo de fin de grado se proponen tres pruebas con escalados a $\frac{1}{2}$, a $\frac{1}{8}$ y a $\frac{1}{16}$ para comprobar el impacto de dicho escalado. Cabe decir que, para el correcto funcionamiento del detector de acciones, el valor de escalado debe ser el mismo tanto para la fase de entrenamiento como para la de validación.

El clasificador implementado debe distinguir entre cinco acciones diferentes: caminar (figura 3.10), correr (figura 3.11), caerse (figura 3.12), sentarse (figura 3.13) y permanecer estático (figura 3.14). En las figuras mencionadas se muestran algunas imágenes de ejemplo pertenecientes a secuencias en las que una persona realiza la acción de interés.

Los vídeos tanto de entrenamiento como de validación han sido proporcionados por el grupo de investigación **GEINTRA**. De entre todos ellos se elegirán para la fase de entrenamiento aquellos en los que se desarrollen la misma clase o actividad de manera repetitiva para obtener la mayor variadad de secuencias de una misma clase, estos vídeos generalmente están grabados para asegurar la existencia de dichas actividades.

Figura 3.10: Fragmento de una secuencia ejemplo de la clase caminar.

Figura 3.11: Fragmento de una secuencia ejemplo de la clase correr.

Figura 3.12: Fragmento de una secuencia ejemplo de la clase caerse.

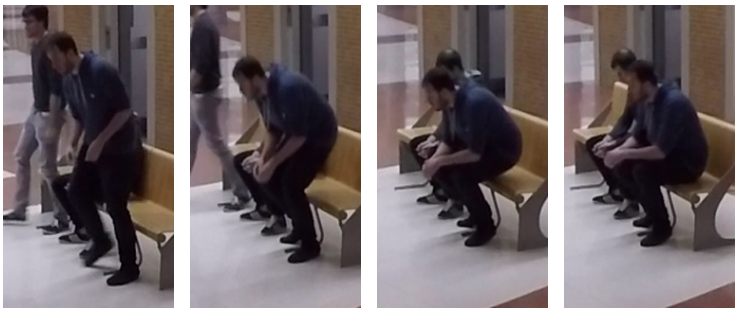


Figura 3.13: Fragmento de una secuencia ejemplo de la clase sentarse.

Figura 3.14: Fragmento de una secuencia ejemplo de la clase permanecer estático.

Por otro lado, para la fase de validación se utilizan vídeos en donde no aparezca una única acción. Generalmente estos vídeos contienen acciones no pactadas previamente.

La máquina de soporte vectorial utilizada para realizar el proceso de clasificación de los vídeos está basada en una SVM linear. Los parámetros que definen dicho clasificador están recogidos en la tabla 2.1 y que son los siguientes:

- $T$=1
- $p$=0.1

Una vez seleccionados los parámetros del clasificador utilizado en este trabajo se proceden a extraer las características mediante ActionBank$^{TM}$ [3, 7] tanto para los videos de entrenamiento como para los de validación. Tal y como se muestra en la figura 3.15 en donde se aprecia que por cada video de entrada se tiene un fichero *banked* y un fichero *featurized*.

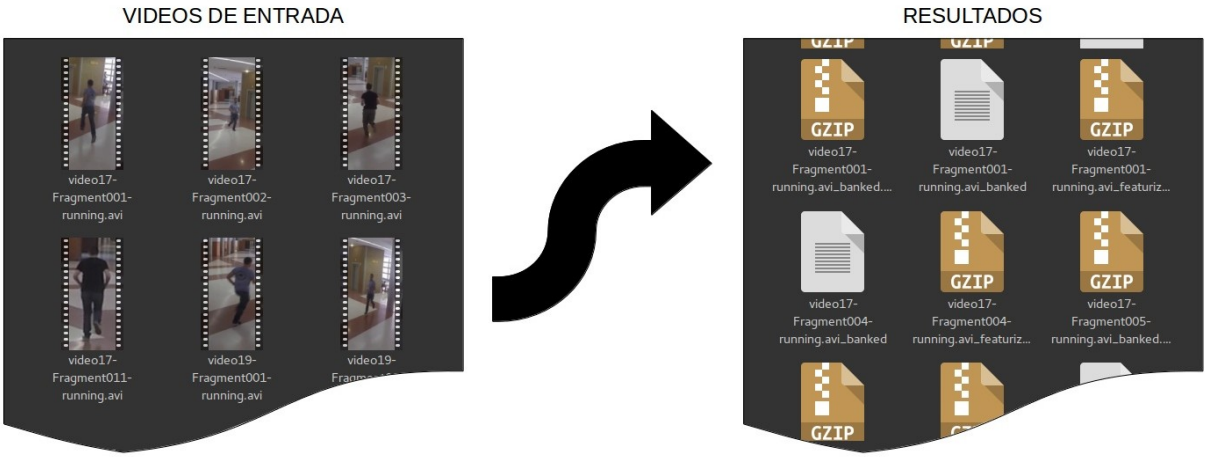



Figura 3.15: Esquema ejemplo de los resultados de Action Bank.

Ya con el clasificador configurado y extraidas las características de los videos que se utilizan se procede a entrenar y validar la SVM utilizada. Para ello se divide ambas fases para facilitar su utilización.

En una primera fase se entrena el clasificador SVM mediante los videos de entrada destinados a dicha función y se obtiene el fichero "*modelfile*" con el cual posteriormente se realizará la clasificación y estimación de los vídeos de entrada de la fase de validación. Este proceso de entrenamiento se ha decidido separarlo del de validación debido a la existencia de más conjuntos de videos de validación que

de entrenamiento, por lo que a diferencia del trabajo de partida, permite realizar las validaciones sin tener que reentrenar el clasificador innecesariamente y por lo tanto el proceso de entrenamiento solo se realiza para cada uno de las pruebas propuestas según cada escalado.

En la fase de validación a partir de los ficheros "*modelfile*" se procede a comprobar el correcto funcionamiento del clasificador previamente entrenado. Para ello se estimarán las clases de los vídeos de entrada destinados a validación y se compararán con el etiquetado realizado anteriormente. Los resultados obtenidos se representan en forma de matriz de confusión y de tablas de tasa de acierto con su correspondiente banda de fiabilidad. Por último se genera un vídeo demostrativo con los resultados de la detección de la actividad.

# Capítulo 4

# Resultados

*Rem tene, verba sequentur (Si dominas el tema, las palabras vendrán solas).*

Catón el Viejo

En este capítulo se presentan los resultados obtenidos en el desarrollo de este trabajo de fin de grado.

Para comenzar, se describen las secuencias de vídeo disponibles tanto para el entrenamiento como para la validación del demostrador desarrollado.

En los apartados siguientes se exponen los resultados obtenidos en lo relativo a la extracción de los segmentos de vídeo (ROIs) tanto de entrenamiento como de test en la fase de detección y extracción de las regiones de interés así como la exposición de las tasas de fallo y acierto del clasificador utilizado para el análisis de la actividad. Cabe destacar que no se presentan resultados de las etapas de detección y seguimiento de personas debido a que dichas etapas no han sido desarrolladas en este trabajo, sino que se han empleado las funciones disponibles, previamente implementadas por Marcos Baptista Ríos, y cuyos resultados pueden consultarse en el trabajo [4].

Finalmente, se presentan los tiempos de procesamiento de cada una de las etapas implicadas en el demostrador. En esta parte se debe tener en cuenta que dichos resultados han sido obtenidos empleando dos ordenadores diferentes: un PC portátil con un procesador *Intel i5* y 4GB de memoria RAM (PC1) y un PC de sobremesa con un procesador *Intel i7* y 8GB de memoria RAM (PC2). Esto ha sido necesario debido a las limitaciones de memoria del PC1 con el que se realizaron las primeras pruebas. Así, atendiendo al esquema de la figura 1.1 el proceso de detección y seguimiento de personas y extracción de regiones de interes es realizado por el PC1 y el proceso de detección de la actividad es realizado por el PC2.

## 4.1 Secuencias de imágenes utilizadas

Los vídeos utilizados tanto para el entrenamiento como para la fase de testeo del demostrador desarrollado han sido proporcionados por el grupo de investigación GEINTRA [6] y han sido grabados con una cámara *GoPro HERO3* [16] de gran angular (lente curvada) utilizando dos configuraciones, la primera con una resolución de $1280 \times 720$ píxeles y 50 *frames* por segundo y una segunda configuración con una resolución de $1920 \times 1080$ píxeles y 59 *frames* por segundo.

Dichos videos facilitados han sido grabados en su totalidad en la zona sur de la Escuela Politécnica Superior de la Universidad de Alcalá de Henares tal y como se muestra en la figura 4.1.

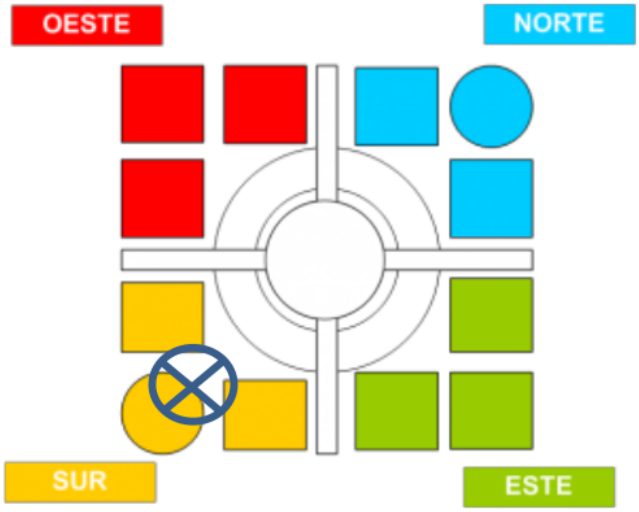



Figura 4.1: Localización de la cámara en la Escuela.

Como se puede observar en la figura 4.2, la región de interés es un pasillo formado por unas escaleras a la derecha situadas entre dos columnas, un ascensor y algunos bancos, los cuales serán movidos y colocados en función del vídeo.

Esta es una de las zonas más concurridas de toda la Escuela, lo que hace que puedan aparezcan en escena personas que no son los actores de los vídeos. Así mismo, la iluminación de la región descrita depende de la hora del día y no está controlada, puesto que se trata de iluminación natural.

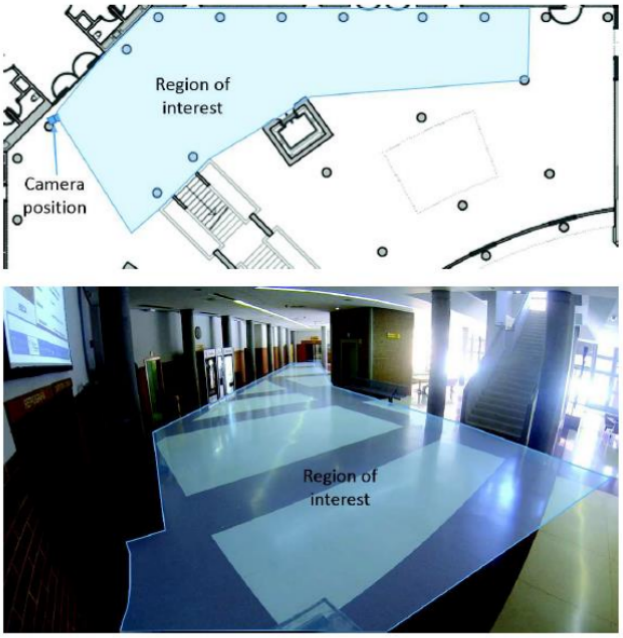



Figura 4.2: Localización de la cámara en la Escuela.

En la tabla 4.1 se muestra información de todos los vídeos proporcionados por el grupo de investigación GEINTRA [6] incluyendo su duración, su resolución e información adicional.

Tabla 4.1: Información proporcionada por el grupo GEINTRA [6].

| Alias del vídeo | Nombre del vídeo | Resolución (Píxeles) | FPS | Duración | Finalidad |
|---|---|---|---|---|---|
| vídeo 1 | GOPR0472.MP4 | 1280 × 720 | 50 | 3:57 | *test* |
| vídeo 2 | GOPR0473.MP4 | 1280 × 720 | 50 | 1:15 | *test* |
| vídeo 3 | GOPR0474.MP4 | 1280 × 720 | 50 | 0:46 | *test* |
| vídeo 4 | GOPR0475.MP4 | 1280 × 720 | 50 | 3:26 | *test* |
| vídeo 5 | GOPR0476.MP4 | 1280 × 720 | 50 | 1:15 | *test* |
| vídeo 6 | GOPR0477.MP4 | 1280 × 720 | 50 | 0:02 | *test* |
| vídeo 7 | GOPR0478.MP4 | 1280 × 720 | 50 | 0:45 | *test* |
| vídeo 8 | GOPR0479.MP4 | 1280 × 720 | 50 | 0:24 | *test* |
| vídeo 9 | GOPR0009.MP4 | 1280 × 720 | 50 | 4:24 | *test* |
| vídeo 10 | GOPR0010.MP4 | 1280 × 720 | 50 | 1:01 | *test* |
| vídeo 11 | GOPR0011.MP4 | 1280 × 720 | 50 | 0:02 | *test* |
| vídeo 12 | GOPR0012.MP4 | 1280 × 720 | 50 | 0:05 | *test* |
| vídeo 13 | GOPR0013.MP4 | 1280 × 720 | 50 | 2:42 | *test* |
| vídeo 14 | GOPR0014.MP4 | 1280 × 720 | 50 | 1:29 | *test* |
| vídeo 15 | GOPR0015.MP4 | 1280 × 720 | 50 | 0:40 | *test* |
| vídeo 16 | GOPR0016.MP4 | 1280 × 720 | 50 | 2:50 | *test* |
| vídeo 17 | GOPR0017.MP4 | 1280 × 720 | 50 | 1:12 | *training* |
| vídeo 18 | GOPR0018.MP4 | 1280 × 720 | 50 | 1:40 | *test* |
| vídeo 19 | GOPR0019.MP4 | 1280 × 720 | 50 | 1:08 | *training* |
| vídeo 20 | GOPRO169.MP4 | 1920 × 1080 | 59 | 1:34 | *test* |
| vídeo 21 | GOPRO170.MP4 | 1920 × 1080 | 59 | 1:05 | *training* |
| vídeo 22 | GOPRO171.MP4 | 1920 × 1080 | 59 | 2:45 | *training* |

De los vídeos mostrados en la tabla 4.1 se extraen las acciones que posteriormente se etiquetan como alguna de las clases posibles: caminar (figura 3.10), correr (figura 3.11), sentarse (figura 3.13), caerse (figura 3.12) y permanecer estático (figura 3.14). Además, debido a que el filtro de Kalman puede proporcionar detecciones erróneas (sombras o reflejos) para los vídeos de validación se proporciona una clase adicional denominada desconocida (ver figura 4.3) para ese tipo de detecciones.

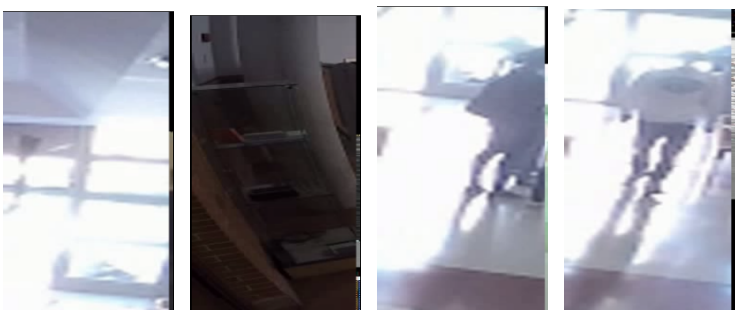


Figura 4.3: Fragmento de una secuencia ejemplo de la clase desconocida.

Para los vídeos destinados a la fase de entrenamiento el etiquetado se hace manualmente tal y como se ha explicado en la sección 3.1.1 y se obtienen los resultados mostrados en la tabla 4.2 los cuales muestran

por cada vídeo, el número de secuencias en las que algún usuario realiza una acción determinada. Además, se muestran el número de ejemplos por cada una de las las clases a detectar, así como como número de ROIs extraídas por cada vídeo.

Tabla 4.2: Acciones extraídas para el entrenamiento

| vídeo | Caminar | Sentarse | Estático | Corriendo | Cayendose | Total |
|---|---|---|---|---|---|---|
| vídeo 17 | 0 | 0 | 0 | 11 | 0 | 11 |
| vídeo 19 | 23 | 0 | 0 | 8 | 0 | 31 |
| vídeo 21 | 23 | 0 | 1 | 0 | 10 | 34 |
| vídeo 22 | 78 | 23 | 42 | 0 | 8 | 151 |
| vídeo totales | 124 | 23 | 43 | 19 | 18 | 227 |

Para los vídeos utilizados en la validación del demostrador, la extracción de las ROIs es automática (ver sección 3.1.2). Además, tal y como se ha explicado, para esta fase se han generado dos conjuntos de datos (*datasets*) diferentes en función de la duración (fija o variable) de los vídeos de acciones. Debido a la diferente duración en cada caso, el número de vídeos extraído para cada clase es diferente en cada *dataset*.

En primer lugar, la tabla 4.3 muestra la información de los datos obtenidos al extraer ROIs de duración variable entre uno y tres segundos. Igual que en el caso anterior, se muestra el número de ROIs extraídas, de cada clase, por cada uno de los vídeos. Para determinar la acción asociada a cada ROI, se ha realizado un etiquetado manual de los mismos.

Tabla 4.3: Acciones extraidas para el *testing* con duración variable.

| vídeo | Caminar | Sentarse | Estático | Corriendo | Cayendose | Desconocido | Total |
|---|---|---|---|---|---|---|---|
| vídeo 1 | 21 | 0 | 0 | 0 | 0 | 0 | 21 |
| vídeo 2 | 0 | 0 | 0 | 6 | 0 | 0 | 6 |
| vídeo 3 | 0 | 0 | 0 | 2 | 0 | 0 | 2 |
| vídeo 4 | 17 | 1 | 3 | 0 | 0 | 0 | 21 |
| vídeo 5 | 9 | 4 | 0 | 1 | 0 | 1 | 15 |
| vídeo 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| vídeo 7 | 1 | 0 | 0 | 0 | 2 | 0 | 3 |
| vídeo 8 | 1 | 0 | 0 | 0 | 1 | 0 | 2 |
| vídeo 9 | 21 | 0 | 0 | 0 | 0 | 23 | 44 |
| vídeo 10 | 1 | 0 | 0 | 0 | 0 | 2 | 3 |
| vídeo 11 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| vídeo 12 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| vídeo 13 | 17 | 0 | 3 | 0 | 0 | 22 | 42 |
| vídeo 14 | 9 | 0 | 0 | 5 | 0 | 27 | 41 |
| vídeo 15 | 2 | 0 | 2 | 3 | 0 | 12 | 19 |
| vídeo 16 | 18 | 0 | 0 | 0 | 0 | 13 | 31 |
| vídeo 18 | 10 | 0 | 0 | 0 | 0 | 9 | 19 |
| vídeo 20 | 4 | 0 | 0 | 0 | 2 | 1 | 7 |
| vídeo total | 131 | 5 | 8 | 17 | 5 | 110 | 276 |

Para el caso de extraer las ROIs con una duración fija de un segundo se obtiene la información mostrada en la tabla 4.4.

Tabla 4.4: Acciones extraidas para el *testing* con duración fija.

| vídeo | Caminar | Sentarse | Estático | Corriendo | Cayendose | Desconocido | Total |
|---|---|---|---|---|---|---|---|
| vídeo 1 | 46 | 0 | 0 | 0 | 0 | 0 | 46 |
| vídeo 2 | 0 | 0 | 0 | 6 | 0 | 0 | 6 |
| vídeo 3 | 0 | 0 | 0 | 3 | 0 | 0 | 3 |
| vídeo 4 | 32 | 3 | 3 | 0 | 0 | 2 | 40 |
| vídeo 5 | 23 | 2 | 0 | 1 | 0 | 1 | 27 |
| vídeo 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| vídeo 7 | 3 | 0 | 0 | 0 | 2 | 0 | 6 |
| vídeo 8 | 1 | 0 | 0 | 0 | 1 | 0 | 2 |
| vídeo 9 | 42 | 0 | 0 | 0 | 0 | 38 | 80 |
| vídeo 10 | 3 | 0 | 0 | 0 | 0 | 3 | 6 |
| vídeo 11 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| vídeo 12 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| vídeo 13 | 35 | 0 | 8 | 0 | 0 | 45 | 88 |
| vídeo 14 | 16 | 0 | 0 | 3 | 0 | 28 | 47 |
| vídeo 15 | 1 | 0 | 4 | 2 | 0 | 13 | 20 |
| vídeo 16 | 47 | 0 | 0 | 0 | 0 | 16 | 63 |
| vídeo 18 | 24 | 0 | 0 | 0 | 0 | 14 | 38 |
| vídeo 20 | 15 | 0 | 0 | 1 | 1 | 0 | 17 |
| vídeo total | 288 | 5 | 15 | 16 | 5 | 160 | 489 |

Tanto la información mostrada en las tablas 4.3 y 4.4 también se emplean para obtener las tasas de acierto en la etapa de detección de acciones en función de la duración de los videos utilizados.

## 4.2 Resultados del extractor de ROI's

Tal y como se ha explicado en la sección 3.1, este módulo se encarga de la detección de personas y su seguimiento en secuencias de vídeo, así como de la extracción de las regiones de interés, que posteriormente se emplearán en la etapa de reconocimiento de actividades.

Como ya se ha comentado, el funcionamiento de este módulo depende de si las ROIs a extraer son para el entrenamiento de la SVM de clasificación de acciones, o para la validación del mismo.

En la figura 4.4 se muestra un fotograma de un vídeo de validación en el que se producen varias detecciones y en la figura 4.5 se observa como a partir de dichas detecciones se extraen las regiones de interés. En el caso de que la finalidad fuese la de entrenamiento el método de obtención varía, pero el resultado final es el mismo al mostrado.

El proceso descrito se repite para todos los vídeos descritos en la tabla 4.1 de tal manera que al final de la extracción de todas las ROIs, tanto de entrenamiento como de validación, se obtiene un *data set* el cual servirá para entrenar y validar el clasificador, además de posibilitar la formación de los vídeos de demostración.


Figura 4.4: Fotograma ejemplo de una detección múltiple de un vídeo destinado a validación.

Figura 4.5: Fotogramas ejemplos de la extracción de las ROI's de una detección multiple de un video destinado a validación.

## 4.3 Resultados de la detección de la actividad

En este apartado se presentan los resultados obtenidos tras el entrenamiento del clasificador SVM tal como se ha explicado en la sección 3.2.

Tal y como se ha comentado en la sección 3.2 Action Bank$^{TM}$ [7] [3] permite ,en cuanto al cálculo de los *features*, un escalado del vídeo de entrada para mejorar los tiempos de cómputo pero en detrimento de la calidad de dichas características calculadas. Para estudiar el impacto de este escalado se proponen tres pruebas, cada una con un escalado distinto ($\frac{1}{2}$, $\frac{1}{8}$ y $\frac{1}{16}$). Estas pruebas se repetirán para los dos conjuntos de vídeos obtenidos en el módulo anterior (ver sección 4.2).

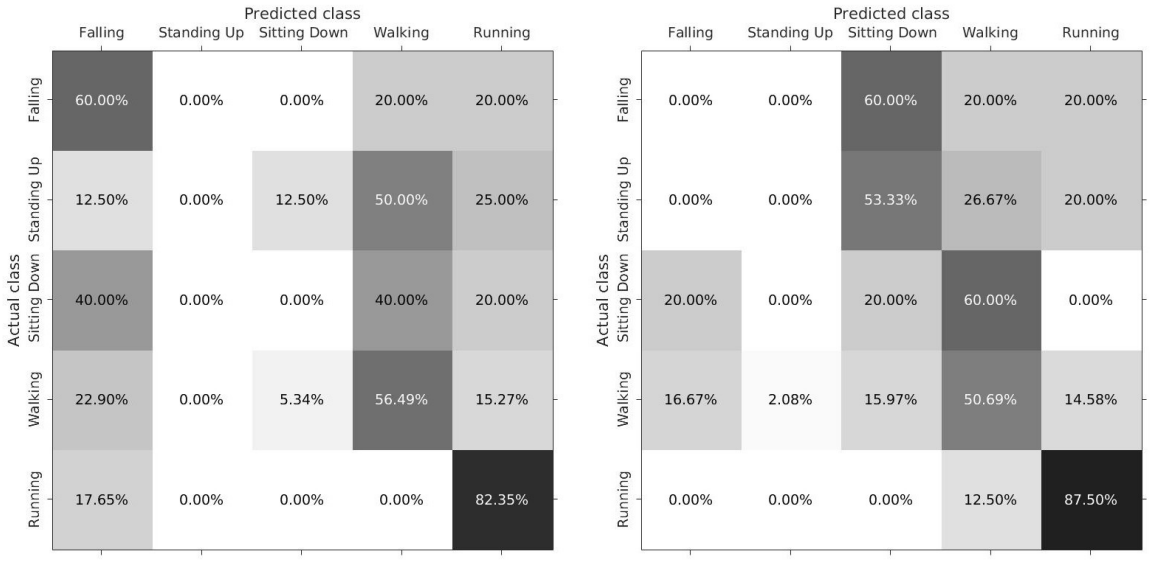
Para comenzar se muestran las matrices de confusión de los diferentes experimentos realizados. Esta matrices proporcionan información acerca de los aciertos y fallos del algoritmo, así como de la confusión

entre clases. El cálculo de las tasas de acierto de cada una de las clases a detectar, así como de la confusión entre clases se realiza mediante el algoritmo 4.1.

**while** *¿quedan etiquetas por leer?* **do**
  **if** *¿La* $etiqueta_{asignada}$ *y la* $etiqueta_{estimada}$ *coinciden?* **then**
    $Tasa_{acierto}++;$
  **else**
    **if** $etiqueta_{estimada} == Clase_1$ **then**
      $Tasa_{Fallo_{Clase_1}}++;$
    **if** $etiqueta_{estimada} == Clase_2$ **then**
      $Tasa_{Fallo_{Clase_2}}++;$
    ⋮
*Cálculo de la* $Tasa_{acierto}$ *en* % ;
*Cálculo de las* $Tasa_{Fallo}$ *de cada clase en* % ;

**Algoritmo 4.1:** Extracción de la tasa de acierto de una clase

Las matrices de confusión mostradas en las figuras 4.6a y 4.6b son las relativas a los resultados obtenidos en la fase de validación del clasificador tanto para vídeos de acciones con longitud variable y fija respectivamente con un escalado de $\frac{1}{16}$.

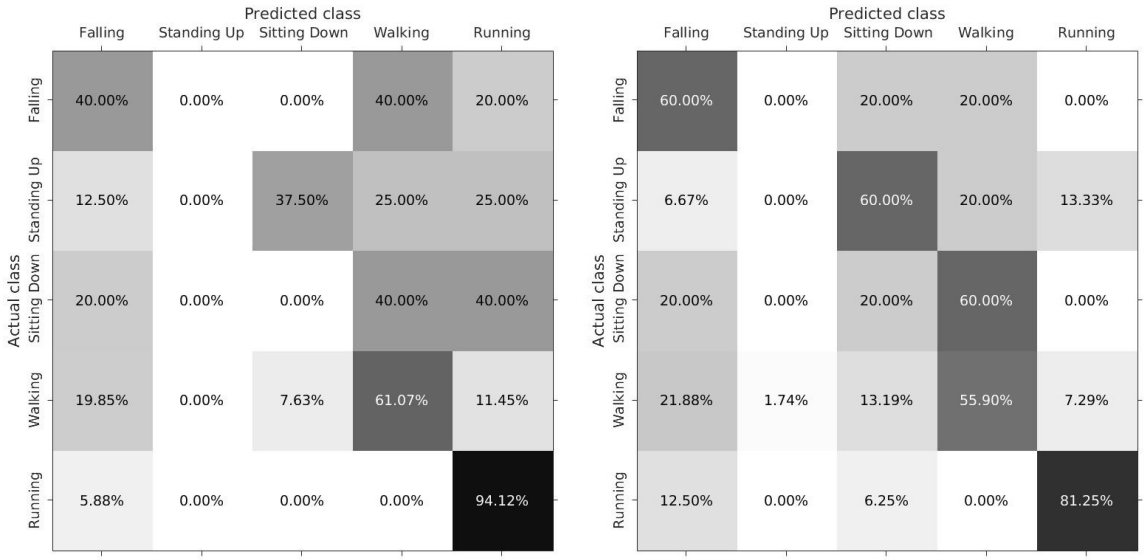



(a) Duración variable.

(b) Duración fija.

Figura 4.6: Matrices de confusión con escalado a $\frac{1}{16}$

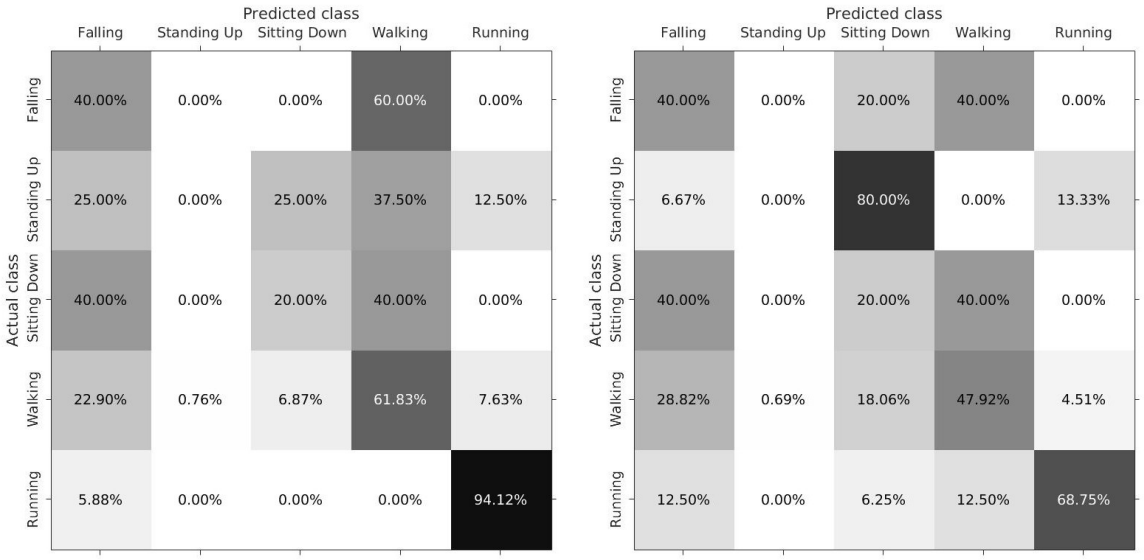
En las figuras 4.7a y 4.7b se muestran las matrices de confusión para los resultados de la fase de validación obtenidos para los vídeos de acciones escalados $\frac{1}{8}$ con duración variable y fija respectivamente.




(a) Duración variable. (b) Duración fija.

Figura 4.7: Matrices de confusión con escalado a $\frac{1}{8}$

En cuanto a las figuras 4.8a y 4.8b se muestran los resultados en forma de matriz de confusión de la fase de validación para los vídeos de acciones que han sido escalados a $\frac{1}{2}$ tanto para duración variable como para duración fija respectivamente.




(a) Duración variable. (b) Duración fija.

Figura 4.8: Matrices de confusión con escalado a $\frac{1}{2}$

A modo de resumen, las tasas de acierto para las diferentes clases, en los experimentos realizados, se presentan en la tabla 4.5, donde además se indica la fiabilidad obtenida de acuerdo a la siguiente expresión:

$$\pm 1{,}96\sqrt{\frac{tasa_{acierto} \times tasa_{fallo}}{numero_{deteccionesClase}}} \tag{4.1}$$

| Clase | Variable y $\frac{1}{16}$ | Fija y $\frac{1}{16}$ | Variable y $\frac{1}{8}$ | Fija y $\frac{1}{8}$ | Variable y $\frac{1}{2}$ | Fija y $\frac{1}{2}$ |
|---|---|---|---|---|---|---|
| Caerse | 60.00 ± 42.94 | 0.00 ± 0.00 | 40.00 ± 42.94 | 60.00 ± 42.94 | 40.00 ± 42.94 | 40.00 ± 42.94 |
| Estático | 0.00 ± 0.00 | 0 ± 0.00 | 0.00 ± 0.00 | 0.00 ± 0.00 | 0.00 ± 0.00 | 0.00 ± 0.00 |
| Sentarse | 0.00 ± 0.00 | 20.00 ± 35.06 | 0.00 ± 0.00 | 20.00 ± 35.06 | 20.00 ± 35.06 | 20.00 ± 35.06 |
| Andando | 56.49 ± 8.49 | 50.69 ± 5.77 | 61.07 ± 8.35 | 55.90 ± 5.73 | 61.83 ± 8.32 | 47.92 ± 5.77 |
| Corriendo | 82.35 ± 18.12 | 87.50 ± 16.21 | 94.12 ± 11.19 | 81.25 ± 19.13 | 94.12 ± 11.19 | 68.75 ± 22.71 |

Tabla 4.5: Tabla comparativa de los resultados de la fase de validación en tasa de acierto ( %) ± banda de fiabilidad ( %) en función del tipo de duración y el escalado del vídeo

Atendiendo a los resultados mostrados anteriormente se observa que de manera general el clasificador no es el más óptimo, ya que las tasas de acierto no son del todo deseables. Si se realiza un análisis clase a clase se puede concluir lo siguente:

- En cuanto a la clase *Caerse*, se puede observar que en ninguna de las pruebas la tasa de acierto supera el 60 % además de que la banda de fiabilidad en todos los casos es demasiado amplia. Exiten dos principales factores que pueden contribuir a ello: la escasez de vídeos existentes de esta clase para la fase de entrenamiento y que las escenas de la fase de validación no están acotadas debido a la extracción automática de ellas.

- Para la clase *Estático* se observa que para todas las pruebas realizadas se obtiene un 0 % de tasa de acierto. Esto es debido principalmente a que el banco de acciones utilizado (Action Bank$^{TM}$ [7] [3]) no es capaz de extraer satisfactoriamente características de una secuencia en la que no exite ningún movimiento.

- Analizando la clase *Sentarse* se observa que los resultados son muy poco satisfactorios, no superandose en ninguna de las clases más del 20 % de tasa de acierto. Si se analizan las matrices de confusión de todas las pruebas realizadas, se aprecia como para esta clase la principal confusión es con las clases de *Caminar* y *Caerese*. Esto es debido a porque las acciones de la fase de validación no están acotadas, por lo que hay fragmentos en los que se realizará la acción de caminar en mayor o menor medida y porque la acción de sentarse es fácilmente confundible con la acción de caerse.

- Para la clase *Caminar* se puede observar que aunque las tasa de acierto no superan más del 62 %, si nos fijamos en las matrices de confusión, la principal confusión se encuentra con la clase *Caerse*. Esto es debido a que en las secuencias utilizadas para la fase de entrenamiento existen pequeños fragmentos en los que puede aparecer la acción de caminar, produciendose así la confusión

- En cuanto a la clase *Correr* es la que mejores resultados obtiene, teniendo como máxima tasa de acierto un 94.12 %

Como último resultado se obtiene el video demostrador de la aplicación. Para conseguirlo se utilizan los resultados de la fase de validación y se asigna un código de colores a las clases siendo estos el azul para la clase *Caerse*, verde para *Estático*, rojo para *Sentarse*, celeste para *Caminar* y rosa para *Correr*. EL resultado final es el que se muestra en la figura 4.9.


Figura 4.9: Fotograma ejemplo del resultado final del demostrador.

## 4.4 Tiempos de ejecución de la aplicación

Por último se procede a analizar los tiempos de ejecución de la aplicación desarrollada en este trabajo de fin de grado, procediendo en primer lugar con el extractor de ROI's y en segundo con el extractor de características Action Bank$^{TM}$ [7] [3].

Analizando la etapa de extracción de ROI's se obtiene que para un *frame* el tiempo necesario para detectar,seguir y extraer la región de intrés es de $432ms$. Este dato se ha obtenido como la media del tiempo de cómputo del análisis "*frame* a *frame*" de cada vídeo proporcionado por el grupo de investigación GEINTRA. Por lo que se puede concluir que para un video de un segundo (50 *frames*) se obtiene un tiempo de cómputo extra de 21.6 segundos.

En cuanto a los tiempos de ejecución a la hora de extraer las caracteristicas mediante Action Bank$^{TM}$ [7] [3] se obtiene que varían en función del escalado. En la tabla 4.6 se muestran los tiempos de análisis de un vídeo de 1 segundo a distintos escalados, cabe resaltar que estos tiempos de ejecución son obtenidos con el PC2 detallado al principio de este capítulo.

| **Escalado a $\frac{1}{2}$** | **Escalado a $\frac{1}{8}$** | **Escalado a $\frac{1}{16}$** |
|---|---|---|
| 6:34 | 2:44 | 0:55 |

Tabla 4.6: Tabla comparativa de los tiempos de ejecución (en minutos y segundos) de Action Bank [7] [3] para un vídeo de 1 segundo

En conclusión atendiendo a los resultados de los tiempos de ejecución obtenidos se presentan los resultados de la tabla 4.7 en los cuales se observan los tiempos globales de ejecución para un vídeo de un segundo. En dicha tabla se puede observar que por el momento no es posible realizar el proceso en tiempo real,

| **Escalado a $\frac{1}{2}$** | **Escalado a $\frac{1}{8}$** | **Escalado a $\frac{1}{16}$** |
|---|---|---|
| 6:56 | 3:06 | 1:17 |

Tabla 4.7: Tabla comparativa de los tiempos globales de ejecución (en minutos y segundos) para un vídeo de 1 segundo

# Capítulo 5

# Conclusiones y líneas futuras

En este capítulo se exponen las conclusiones a las que se han llegado tras la realización de este trabajo, así como las lineas futuras que se deberían seguir tras la realización del mismo.

## 5.1 Conclusiones

El trabajo de fin de grado desarrollado ha consistido en la implementación de un demostrador para la detección de personas y el análisis de su actividad en imágenes de color. Para ello se ha tenido que recurrir a diversas técnicas tanto para la detección de personas mediante la obtención de descriptores HOG con la técnica de ventana deslizante y la posterior utilización de un clasificador SVM, el seguimiento de las mismas con ayuda de un banco de filtros de Kalman y la extracción automática de las secuencias mediante la utilización de las regiones de interés (ROIs) y el posterior uso de dichas secuencias para detectar las acciones utilizando descriptores de ActionBank$^{TM}$ [3, 7] para la extracción de características y el uso de un otro clasificador SVM para el análisis de la actividad realizada.

Se ha analizado el extractor de regiones de interés y aunque no se puede extraer ninguna conclusión en cuanto al número de secuencias obtenidas, ya que estas están definidas por las detecciones del sistema desarrollado en [4], si se puede observar un número relativamente alto de secuencias marcadas como desconocidas que tienen que ser desechadas manualmente por el usuario. Además se observa un gran desequilibrio en cuanto a las acciones obtenidas tanto para la fase de entrenamiento como para la fase de validación lo cual tendrá un impacto sobre la calidad del clasificador SVM.

Además, se ha evaluado la calidad del clasificador utilizado mediante matrices de confusión y tablas de tasas de acierto y bandas de fiabilidad con el objetivo de fijar tanto el escalado como la duración óptima de las secuencias de entrada proporcionadas al clasificador SVM. En base a la tabla 4.5 se puede concluir que los mejores resultados se consiguen para una duración fija de un segundo y un escalado de $\frac{1}{8}$ y para una duración variable de entre uno y tres segundos con un escalado de $\frac{1}{2}$.

Atendiendo a los resultados obtenidos para los tiempos de ejecución la opción más rentable en cuanto a calidad obtenida y tiempo utilizado sería la de duración fija y escalado a $\frac{1}{8}$.

Por último cabe destacar que las tasas de acierto obtenidas son inferiores a las esperadas, especialmente para la clase permanecer estático. De las pruebas realizadas se deduce que los errores para esa clase se deben a que ActionBank$^{TM}$ [3, 7] no estaría diseñado para la extracción de características para secuencias con ausencia de movimiento. Además, la diferencia en cuanto al número de vídeos de entrenamiento para las diferentes clases consideradas, también tiene un efecto negativo en los resultados de la clasificación.

Sin embargo, a pesar de los problemas descritos, los objetivos planteados en este TFG se han alcanzado, permitiendo desarrollar un sistema que realiza de forma autónoma la detección y seguimiento de personas, extracción de ROIs de duración fija o variable, y reconocimiento de actividades, así como la visualización de resultados.

## 5.2 Líneas futuras

Para finalizar, se dará paso a plantear unas posibles líneas futuras obtenidas tras la realización de este trabajo de fin de grado:

- **Estudio en la infulencia de la distancia al hierplano en la detección de personas para la eliminación de posibles resultados erroneos.** Debido a la presencia de un campo proporcionado por el sistema desarrollado en [4] que devuelve la distancia al hiperplano del clasificador SVM utilizado según si se detecta persona o no, cabría la posibilidad de estudiar la relación que tiene este parámetro con el hecho de que la clasificación sea o no errónea y así poder descartar automáticamente esas detecciones no deseadas.
- **Aumento de la base de datos de los vídeos de entrada.** Tras los resultados poco satisfactorios del clasificador utilizado para la detección de actividad, se propone aumentar la cantidad de vídeos disponibles en la base de datos y estudiar si se produce una mejora en los resultados obtenidos.
- **Ampliación en la finalidad de la aplicación para la detección de comportamientos anómalos.** Una vez probada esta aplicación en la detección de la actividad realizadas por personas sería interesante comprobar la efectividad de dicho software cuando se traslada a otros campos de estudio como la detección de anomalías en secuencias de vídeo.

# Bibliografía


[1] https://en.wikipedia.org/wiki/Histogram_of_oriented_gradients [Último acceso 21/junio/2017].

[2] https://en.wikipedia.org/wiki/Support_vector_machine [Último acceso 21/junio/2017].

[3] J. J. Corso, "Action bank: A high-level representation of activity in video," in *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, ser. CVPR '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 1234–1241. [Online]. Available: http://dl.acm.org/citation.cfm?id=2354409.2354937

[4] M. B. Ríos, "Detección y caracterización de personas en espacios inteligentes con cámaras de color," 2015.

[5] V. B. Arévalo, "Diseño, generación y anotación de base de datos de secuencias de imágenes en interiores para aplicaciones de video-vigilancia," 2016.

[6] "Página web del grupo de investigación geintra," http:/http://www.geintra-uah.org/ [Último acceso 30/noviembre/2016].

[7] "Action bank: A high-level representation of activity in video," http://www.cse.buffalo.edu/~jcorso/r/actionbank/ [Último acceso 22/mayo/2017].

[8] C. M. García, "Reconocimiento de actividad humana en secuencias de vídeo," 2015.

[9] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, June 2005, pp. 886–893 vol. 1.

[10] http://www.itl.nist.gov/div898/handbook/pmc/section3/pmc324.htm [Último acceso 21/junio/2017].

[11] http://www.itl.nist.gov/div898/handbook/pmc/section3/pmc324.htm [Último acceso 21/junio/2017].

[12] http://www.mdpi.com/1424-8220/14/4/6474/htm [Último acceso 10/septiembre/2017].

[13] "Página con documentación acerca de la librería OpenCV," http://docs.opencv.org/ [Último acceso 30/noviembre/2016].

[14] Itseez, "Open source computer vision library," https://github.com/itseez/opencv, 2015.

[15] *The OpenCV Reference Manual*, 2nd ed., Itseez, April 2014.

[16] "Página oficial de la cámra gopro," https://es.shop.gopro.com/EMEA/cameras/?gclid=CIXK49qYi8gCFUe3Gwod1usCKg [Último acceso 02/septiembre/2017].

# Apéndice A

# Pliego de condiciones

A continuación se presentan los requisitos mínimos necesarios que se debe cumplir, tanto de hardware como de software, para la correcta ejecución del sistema desarrollado en este trabajo de fin de grado.

## A.1 Requisitos de Hardware

- Procesador de 64 bits con 4 núcleos
- 8GB de memoria RAM o superior
- Al menos 12GB de memoria libre en el disco duro
- Cámara de vídeo GoPro Hero3 o similar

## A.2 Requisitos de Software

- Sistema operativo *Linux Ubuntu 16.04.1 LTS*
- Libreria *OpenCV 2.4.9*
- Libreria *Python 2.7.12*
- Software *Matlab 2016b*
- Software *TeXmaker 4.4.1*
- Compilador GNU GCC

# Apéndice B

# Presupuesto

## B.1 Costes de equipamiento

- Equipamiento hardware utilizado

| **Concepto** | **Cantidad** | **Coste Unitario** | **Subtotal(€)** |
|---|---|---|---|
| Asus GL552VW-DM149 Core i5 4GB 1TB GTX960M 15.6" | 1 | 790€ | 790€ |
| Ordenador sobremesa procesador *Intel i7* y 8GB de RAM | 1 | 950€ | 950€ |
| GoPro Hero3 | 1 | 350€ | 350€ |
| **Coste total HW** | | | 2090€ |

- Recursos software utilizados

| **Concepto** | **Cantidad** | **Coste Unitario** | **Subtotal(€)** |
|---|---|---|---|
| *Ubuntu 16.04.1 LTS* | 1 | 0€ | 0€ |
| Librería *OpenCV 2.4.9* | 1 | 0€ | 0€ |
| Librería *Python 2.7.12* | 1 | 0€ | 0€ |
| Licencia *Matlab* educativa | 1 | 500€ | 500€ |
| Software *Latex* | 1 | 0€ | 0€ |
| **Coste total SW** | | | 500€ |

## B.2 Costes de mano de obra

| **Concepto** | **Cantidad** | **Coste Unitario** | **Subtotal(€)** |
|---|---|---|---|
| Desarrollo SW | 250 | 60€/hora | 15000€ |
| Mecanografiado del documento | 50 | 15€/hora | 750€ |
| **Coste total mano de obra** | | | 15750€ |

## B.3 Coste total

| **Concepto** | **Subtotal(€)** |
|---|---|
| Equipamiento hardware | 2090€ |
| Recursos software | 500€ |
| Mano de obra | 15750€ |
| **Coste total** | 18340€ |

# Apéndice C

# Manual de usuario

Este apéndice ofrece un manual tanto para la instalación como para el uso de la aplicación desarrollada en este trabajo de fin de grado.

Dicho trabajo hace uso de diversos programas desarrollado en *C++*, *Matlab* y *Python*.

## C.1 Instalación de librerias en entorno Linux

### C.1.1 Instalación del entorno Python + Scipy

En la mayoria de distribuciones de linux viene ya instalado de serie. Para comprobar si se dispone de él y de que versión, se realiza desde un terminal lo siguiente:

```
$ python -V
```

En caso de obtener un error, significa que no disponemos de *Python* en el sistema. Para obtenerlo, se realiza lo siguiente (sustituyendo "*ver" por la versión de *Python* que se desea instalar):

```
$ sudo apt-get install python*ver
```

Una vez que se dispone de *Python*en el sistema, es necesario para ejecutar el software realizado en este TFG, ScyPy. Se trata de un software open-source, para Python, con el que realizar tareas matemáticas, científicas y de ingeniería.

Algunos de los cores más importantes que incorpora son:

- Numpy. Para la operación con arrays y matrices N-dimensionales.
- SciPy library. Librería para computación científica.
- Matplotlib. Se trata de una librería para la representación gráfica de datos.

Para realizar la instalación, se realiza lo siguiente:

```
$ sudo apt-get install python-numpy python-scipy python-matplotlib ipython
ipython-notebook python-pandas python-sympy python-nose
```

Tras ello se dispone, tanto de Python, como del software SciPy, correctamente instalados.

### C.1.2 Instalación de la herramienta FFmpeg

FFmpeg es una herramienta libre y multiplataforma, que sirve para grabar, convertir y hacer streaming tanto de audio como de vídeo.

Para instalar FFmpeg en un sistema Linux, hay que realizar lo siguiente en una consola de terminal:

```
$ sudo apt-add-repository ppa:mc3man/trusty-media
$ sudo apt-get update
$ sudo apt-get install ffmpeg gstreamer0.10-ffmpeg
```

Una vez ejecutados los comandos anteriores, se dispone de FFmpeg instalado y listo para ser usado.

### C.1.3 Instalación de las librerías OpenCV

En este apartado, se explica el proceso de instalación de la librería de visión artificial OpenCV cuyas principales características son:

- Se trata de una librería open source de visión artificial y machine learning, escrita en C++.
- Está publicada bajo licencia BSD.
- Dispone de interfaces en C++, C, Python, Java y MATLAB.
- Es multiplataforma, soportando Windows, Linux, Android y Mac OS.
- Dispone de más de 2500 algoritmos optimizados. Estos algoritmos pueden ser empleados en tareas como detección y reconocimiento de objetos, tracking de objetos en movimiento, reconocimiento de escenarios para emplearlos en realidad aumentada, etc.

A continuación se indican los pasos a seguir para su instalación. El primero de ellos consiste en descargar la versión estable más reciente, desde su web oficial http://opencv.org/downloads.html. En este caso se procede a descargar la versión 3.0.0 para Linux.

Antes de comenzar el proceso de instalación, es necesaria la instalación de ciertas dependencias. Ejecutando lo siguiente en un terminal, se instalan las dependencias obligatorias (en caso de que no estuvieran instaladas):

```
$ sudo apt-get install build-essential
$ sudo apt-get install cmake git libgtk2.0-dev pkg-config libavcodec-dev libavformat-dev libswscale-dev
```

Una vez se han instalado las dependencias necesarias y se ha descargado la última versión de OpenCV, se procede a instalarla. En primer lugar hay que descomprimir la librería:

```
$ cd ~/Descargas/
$ unzip opencv-3.0.0.zip
```

Tras ello, se compila el código fuente de OpenCV, empleando CMake, de la siguiente forma:

```
$ cd ~/Descargas/opencv-3.0.0/
$ mkdir release
$ cd release
$ cmake -D CMAKE_BUILD_TYPE=RELEASE -D CMAKE_INSTALL_PREFIX=/usr/local ..
```

Con el código fuente ya compilado se instala OpenCV ejecutando las siguientes líneas:

```
$ cd ~/Descargas/opencv-3.0.0/release/
$ make
$ sudo make install
```

Al finalizar la instalación se localiza, si es necesario, donde se encuentran instalados los ficheros de OpenCV. Para ello desde un terminal se realiza lo siguiente:

```
$ pkg-config --cflags opencv -I/usr/local/include/opencv -I/usr/local/include
```

También, desde un terminal, se localizan las librerías y el directorio en el que están instaladas:

```
$ pkg-config --libs opencv -L/usr/local/lib -lopencv_shape -lopencv_stitching -lopencv_objdetect
-lopencv_superres -lopencv_videostab -lopencv_calib3d -lopencv_features2d -lopencv_highgui
-lopencv_videoio -lopencv_imgcodecs -lopencv_video -lopencv_photo -lopencv_ml -lopencv_imgproc
-lopencv_flann -lopencv_core -lopencv_hal
```

Una vez conocida la localización de los ficheros y de las librerías de OpenCV, hay que indicar al sistema donde se encuentran éstos. Para ello, en primer lugar hay que crear un archivo */etc/ld.so.conf.d/opencv.conf*, en el que se escriben los directorios donde se encuentran las librerías de OpenCV. En este caso, se puede realizar de la siguiente forma:

```
$ sudo gedit /etc/ld.so.conf.d/opencv.conf
```

En la ventana de gedit, se escribe el directorio anteriormente localizado: */usr/local/lib*. Tras ello, en un terminal se ejecutará lo siguiente:

```
$ sudo ldconfig -v
```

Finalmente, se incluyen las rutas de los archivos ".so" de OpenCV en *LD_LIBRARY_PATH*. Ejecutando en un terminal lo siguiente:

```
$ export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib
```

Siguiendo los pasos hasta aquí, se dispone de la última versión de la librería OpenCV, instalada y configurada en el sistema.

### C.1.4 Instalación de la librería LIBSVM

LIBSVM es un software que permite implementar SVM (Support Vector Machines), para clasificación, regresión y estimación. Incluye diferentes formulaciones de SVM, kernels y soporte para clasificación multiclase, entre otras muchas características.

Además permite ser usada tanto en sistemas Windows como Linux, con interfaces en numerosos lenguajes (C++, Java, Python, R, MATLAB, etc). Para emplear LIBSVM, lo primero que hay que hacer es descargarla desde la web oficial https://www.csie.ntu.edu.tw/~cjlin/libsvm/.

Una vez descargada y descomprimida, se puede comprobar que incluye diferentes ficheros. Algunos de los más importantes son:

- README. Fichero con información sobre instalación y uso de la librería.
- "svm.h" y "svm.cpp". Se trata de los ficheros que habrá que incluir en un proyecto si se desea usar LIBSVM (source Code).
- "svm-train.c" y "svm-predict.c". Ficheros de ejemplo, para realizar las tareas de TRAIN y de TEST con SVM.

Con la librería ya descargada, se puede incluir ésta en el proyecto que se desee.

### C.1.5 Instalación del Action Bank

Action Bank es un descriptor para la representación de actividades de alto nivel en vídeos. El código fuente, puede ser descargado desde la página web oficial http://www.cse.buffalo.edu/~jcorso/r/actionbank/.

Necesita de las siguientes dependencias para funcionar:

- Python v2.7.
- Numpy/Scipy.
- FFmpeg.
- Shogun toolbox (únicamente si se quiere usar el código para clasificación que incluye).

Todas esas dependencias han sido previamente instaladas si se han seguido los pasos indicados en apartados anteriores. Por lo tanto, para emplearlo, basta simplemente con descomprimirlo.

El proceso de descarga y descompresión puede realizarse en un terminal, de la siguiente manera:

```
$ cd "$HOME"
$ wget http://www.cse.buffalo.edu/~jcorso/extdelivery/actionbank_v1_0.tar.bz
$ tar xjf actionbank_v1_0.tar.bz
```

Siguiendo estos pasos, se puede hacer uso del Action Bank en el proyecto.

## C.2 Manual

En esta sección se muestra la forma de utilización del sistema desarrollado. Se indican los comandos que se deben emplear y las salidas que se obtienen, para un caso de ejemplo.

### C.2.1 Compilación y creación de los ejecutables

Para construir los ejecutables utilizados es necesario compilar las librerías utilizadas para cada uno de ellos. Todos los ficheros *MAKEFILE* son proporcionados, por lo que basta con proceder como se indica a continuación para cada uno de los ejecutables utilizados:

```
$ make -B
```

### C.2.2 Procedimiento para la extracción de las secuencias

Para extraer las secuencias de las imagenes que van a ser analizadas por el clasificador se debe diferenciar entre si la finalidad de ellas es la de entrenamiento o la de validación. Esto es debido a que los videos de entrenamiento se extraen de manera manual para asegurar que las acciones estén acotadas, mientras que para la fase de validación dicha extracción es automática.

### C.2.2.1 Extracción de las secuencias de entrenamiento

Para la extracción de estas secuencias se hace uso de la aplicación de *Matlab* desarrollada en [5]. Al iniciarse la aplicación aparece una ventana emergente como la mostrada en la figura C.1 donde se selecciona el directorio de trabajo. En este caso a modo de ejemplo se selecciona el video 22 como directorio de trabajo.

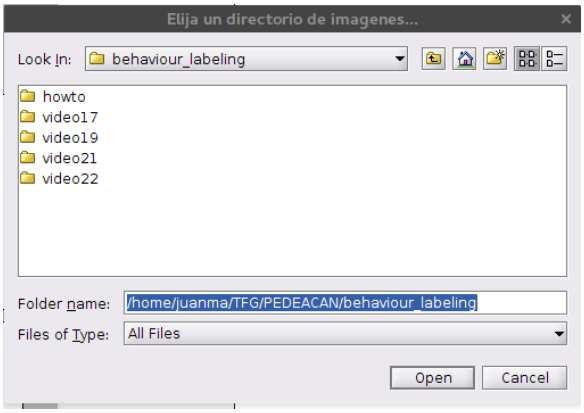



Figura C.1: Ventana emergente inicial de la aplicación desarrollada en [5].

Una vez elegido el directorio, se procede a extraer los *frames* del video contenido en él si no se encuentran disponibles con anterioridad. A continuación, la aplicación muestra un interfaz como el de la figura C.2 donde se pude configurar los *frames* no consecutivos máximos entre los que se realiza la interpolación de las medidas, en este caso se han seleccionado 5 *frames*.

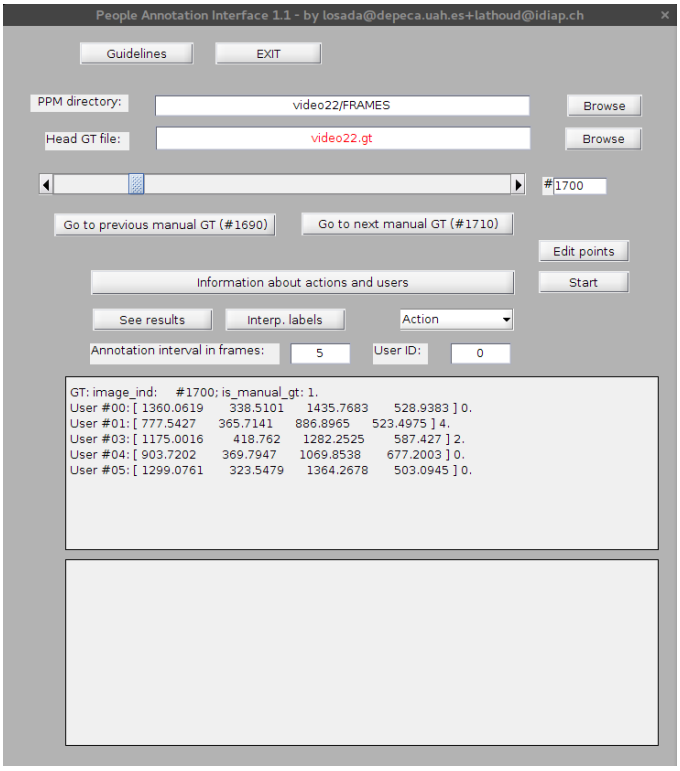



Figura C.2: Interfaz de la aplicación desarrollada en [5].

Hecha esta configuración se selecciona el botón *start* y se procede a realizar el etiquetado manual mediante el desplegable de acciones y seleccionando la esquina superior izquierda y la inferior derecha de

cada persona que aparezca en el *frame* tal y como se muestra en la figura 3.4. Esto se repite sucesivamente con todos los frames y se obtiene un fichero de "*Ground Truth*" que tiene el formato que se muestra en la figura C.4. Dicho formato es el mostrado en la figura C.3 donde:

- **FFFFFF:** es el número de imagen dentro de la secuencia. Este valor es especialmente importante en los casos en que no se etiquetan todas las imágenes, sino algunas de ellas.

- **X:** indica si el etiquetado de esa imagen es manual (1) o automático (0) (mediante interpolación entre las etiquetas manuales).

- **Número de usuario, coordenadas de los puntos etiquetados y acción realizada por el usuario:** a continuación, por cada usuario etiquetado en la imagen, aparecen 6 valores. El primero identifica al usuario ($UID_0$), los siguientes 4 corresponden a las coordenadas (en píxeles) de los puntos etiquetados sobre la imagen y el último valor corresponde al identificador de la acción realizada ($ACT_0$).

FFFFFF X $UID_0$ $PAX_0$ $PAY_0$ $PBX_1$ $PBY_1$ $ACT_0$... $UID_i$ $PAX_i$ $PAY_i$ $PBX_i$ $PBY_i$ $ACT_i$

Figura C.3: Formato del fichero "*Ground Truth*" devuelto por la aplicación desarrollada en [5].

```
000001 1 0000  1204.4430  334.4295  1332.7234  500.3741 0 0001  825.9107  360.2733  920.5438  475.8904 4
000002 0 0000  1204.9103  333.8250  1332.4897  500.5252 0 0001  825.4434  360.4244  919.8428  475.7393 4
000003 0 0000  1205.3777  333.2204  1332.2561  500.6764 0 0001  824.9761  360.5756  919.1418  475.5881 4
000004 0 0000  1205.8450  332.6159  1332.0224  500.8275 0 0001  824.5087  360.7267  918.4408  475.4370 4
000005 0 0000  1206.3123  332.0114  1331.7888  500.9786 0 0001  824.0414  360.8778  917.7398  475.2859 4
000006 0 0000  1206.7797  331.4068  1331.5551  501.1298 0 0001  823.5741  361.0290  917.0389  475.1347 4
000007 0 0000  1207.2470  330.8023  1331.3215  501.2809 0 0001  823.1068  361.1801  916.3379  474.9836 4
000008 0 0000  1207.7143  330.1978  1331.0878  501.4320 0 0001  822.6394  361.3312  915.6369  474.8325 4
000009 0 0000  1208.1817  329.5932  1330.8542  501.5832 0 0001  822.1721  361.4824  914.9359  474.6813 4
000010 1 0000  1208.6490  328.9887  1330.6205  501.7343 0 0001  821.7048  361.6335  914.2349  474.5302 4
000011 0 0000  1208.4387  328.5806  1329.5690  501.7343 0 0001  822.3357  361.7695  915.0761  475.4823 4
000012 0 0000  1208.2284  328.1726  1328.5175  501.7343 0 0001  822.9666  361.9055  915.9173  476.4345 4
000013 0 0000  1208.0181  327.7645  1327.4661  501.7343 0 0001  823.5975  362.0416  916.7585  477.3866 4
000014 0 0000  1207.8078  327.3565  1326.4146  501.7343 0 0001  824.2284  362.1776  917.5997  478.3388 4
000015 0 0000  1207.5975  326.9484  1325.3631  501.7343 0 0001  824.8592  362.3136  918.4408  479.2909 4
000016 0 0000  1207.3872  326.5403  1324.3116  501.7343 0 0001  825.4901  362.4496  919.2820  480.2430 4
000017 0 0000  1207.1769  326.1323  1323.2601  501.7343 0 0001  826.1210  362.5856  920.1232  481.1952 4
000018 0 0000  1206.9666  325.7242  1322.2087  501.7343 0 0001  826.7519  362.7217  920.9644  482.1473 4
000019 0 0000  1206.7563  325.3162  1321.1572  501.7343 0 0001  827.3828  362.8577  921.8056  483.0995 4
000020 1 0000  1206.5460  324.9081  1320.1057  501.7343 0 0001  828.0137  362.9937  922.6468  484.0516 4
```

Figura C.4: Ejemplo del fichero "*Ground Truth*" devuelto por la aplicación desarrollada en [5].

Obtenido el fichero "*Ground Truth*" se procede a extraer las regiones de interes marcadas. Para ello se ejecuta el programa desarrollado en *C++ Gt2ActionVideo* con la siguiente línea de comandos:

```
$ ./Gt2ActionVideo -f -u -d -l "GtFile".gt -v "VideoFile".MP4
```

Con ello se consiguen por cada video de entrenamiento tres videos extras y son nombrados con el formato *"Video_extraccion""Alteracion"Fragment"número_de_fragmento""Acción"*, todos guardados en formato *avi*, con una estructura de archivos como la mostrada en la figura C.5. A modo de ejemplo un posible video puede ser *video22-FilpFragment001-walking.avi*.

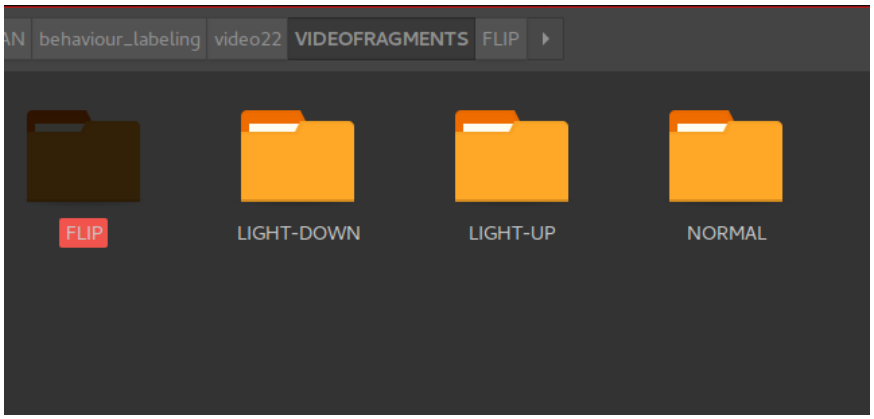



Figura C.5: Ejemplo de la estructura de archivos donde se guardan los videos extraidos.

Terminado este proceso se deben separar los videos por acciones en una estructura de archivos mostrada en la figura C.6.

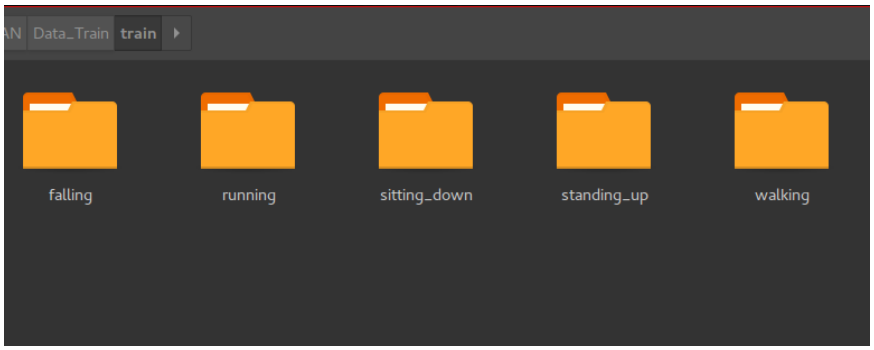



Figura C.6: Ejemplo de la estructura de archivos donde se guardan los videos destinados a entrenamiento.

#### C.2.2.2 Extracción de las secuencias de validación

Para extraer las secuencias de validación se hace uso de tres programas desarrollados en *C++*. El primero de ellos se llama *Video2ImgConverter* y se utiliza para obtener los *frames* del video del que se quieren extraer las acciones. Para ejecutarlo basta con abrir un terminal y teclear la siguiente línea:

```
$ ./Video2ImgConverter "Video".MP4
```

Una vez extraidos todos los frames se debe ejecutar el programa *peopledetect* el cual se encarga de la detección y seguimiento de las personas. Para ejecutarlo simplemente se teclea en un terminal la línea:

```
$ ./peopledetect -s -n ScoringFiles/"Video" -l "Video"/Data.list
```

Para más información sobre los parámetros de este ejecutable basta con introducir el comando:

```
$ ./peopledetect -h
```

Terminada la ejecución del programa hay que extraer las secuencias de video. Para ello se utiliza el programa *VideoSorter* y basta con ejecutar la línea:

```
$ ./VideoSorterScript -l "Video"
```

Para más información sobre los parámetros de este progrma introducir la línea:

```
$ ./VideoSorterScript -h
```

Debido a la complejidad en la ejecución de estos programas y de la estructura de direcctiorios que se necesita. Se ha creado un script que configura todo el entorno y lanza los programas en cadena, por lo que todo lo anteriormente descrito se puede ejecutar mediante la línea de comando:

```
$ ./ProgramScript
```

Quedando al final un directorio llamado *storing* donde se guardan todos los vídeos de validación y un fichero de texto con las estadísticas de los vídeos extraidos, tal y como se muestra en la figura **??**.

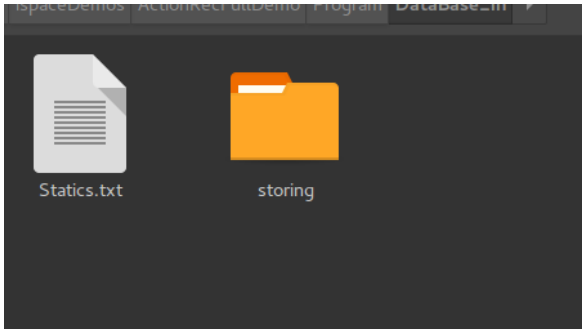



Figura C.7: Ejemplo de la estructura de archivos donde se guardan los videos extraidos para validación.

Para realizar la fase de validación se deben separar los videos por clases, quedando una estructura como la de la figura C.8

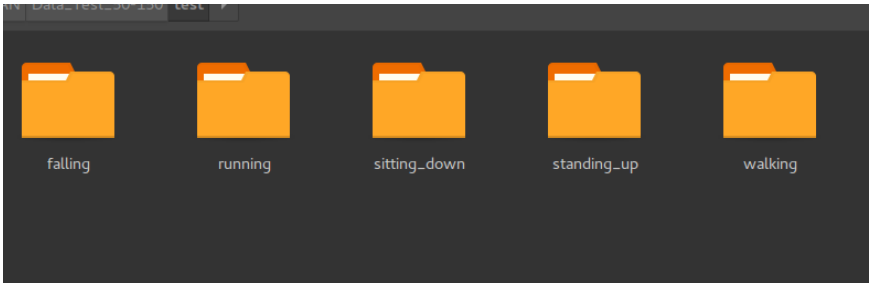



Figura C.8: Ejemplo de la estructura de archivos donde se guardan los videos de validación.

### C.2.3 Procedimiento del sistema de reconocimiento de la actividad humana

Una vez que se disponen de las secuencias que se quieren pasar al sistema de reconocimiento de actividad humana, hay que organizarlas de una forma determinada (si no lo están ya). Se requiere que haya una carpeta con el nombre de la base de datos elegida (por ejemplo “bd_in”) y dentro de ella dos directorios, “train” y “test”. Dentro de cada uno de ellos tiene que haber una carpeta por cada clase (acción a reconocer) de la base de datos. Finalmente dentro de cada una de éstas, deben estar las secuencias de vídeo (generadas por ejemplo de la forma indicada en la sección C.2.2).

Tras ello, se ejecuta en la línea de comandos el siguiente código:

```
$ cd /home/usuario/actionbank_v1_0/code
$ python actionbank.py -c 8 -g 8 /home/usuario/bd_in /home/usuario/bd_procesada
```

En la figura C.9, se muestra un ejemplo de las entradas y salidas al ejecutar el código anterior.

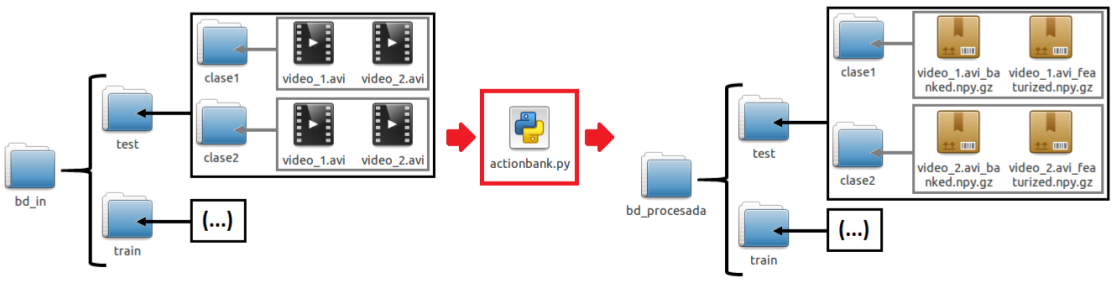



Figura C.9: Ejemplo de E/S del código de ?actionbank.py? indicado.

Tras la ejecución se ha tenido que generar una pareja de ficheros "_banked.npy.gz" y "_featurized.npy.gz", por cada vídeo de entrada (Figura C.9). Únicamente son necesarios para las siguientes etapas los "_banked.npy.gz". Una vez que se dispone de los descriptores (generados por el Action Bank), hay que transformarlos a formato texto, para que la siguiente etapa pueda tratar con ellos. Para ello, hay que ejecutar lo siguiente:

```
$ python ab_txt.py bd_procesada
```

Después de realizar estos pasos, ahora hay que diferenciar entre si se quiere entrenar el clasificador SVM o validarlo. Para la priemra opción hay que ejecutar la siguiente línea:

```
$ ./Train _bd_procesada_Train
```

Con esto se generan los siguientes archivos:

- Fichero con los datos de entrenamiento: "svm_trainfile".
- Modelo entrenado del sistema: "svm_modelfile".

Para realizar la fase de validación se ejecuta la siguiente línea:

```
$ ./Test _bd_procesada_Test
```

Tras ello se obtienen los ficheros mostrados a continuación:

- Fichero con los datos de validación "svm_testfile".
- Fichero con las salidas: "svm_outputfile".
- Ficheros con los resultados del proceso: "svm_resultados" y "svm_resultados2".

A continuación, en la figura C.10 se presenta el contenido del fichero de "svm_resultados" ejemplo, el cual se utiliza para la reconstrucción del vídeo demostrador. En la figura C.11 se muestra un ejemplo del fichero "svm_resultados2" el cual se utiliza para la generación de las matrices de confusión.

```
.....................................................................
CLASES DETECTADAS
.....................................................................
1 -> falling
2 -> standing_up
3 -> sitting_down
4 -> walking
5 -> running
.....................................................................
RESULTADOS OBTENIDOS
Formato -> video * clase_real * clase_estimada
.....................................................................
Data_test_50-150_Out2/test/falling/video7falling01.avi_banked * 1 * 4
Data_test_50-150_Out2/test/falling/video20falling06.avi_banked * 1 * 4
Data_test_50-150_Out2/test/falling/video20falling04.avi_banked * 1 * 4
Data_test_50-150_Out2/test/falling/video8falling01.avi_banked * 1 * 1
Data_test_50-150_Out2/test/falling/video7falling02.avi_banked * 1 * 1
Data_test_50-150_Out2/test/standing_up/video4standing_up04.avi_banked * 2 * 4
Data_test_50-150_Out2/test/standing_up/video15standing_up18.avi_banked * 2 * 4
Data_test_50-150_Out2/test/standing_up/video15standing_up17.avi_banked * 2 * 3
Data_test_50-150_Out2/test/standing_up/video13standing_up15.avi_banked * 2 * 1
Data_test_50-150_Out2/test/standing_up/video13standing_up02.avi_banked * 2 * 3
Data_test_50-150_Out2/test/standing_up/video4standing_up17.avi_banked * 2 * 4
Data_test_50-150_Out2/test/standing_up/video4standing_up20.avi_banked * 2 * 5
Data_test_50-150_Out2/test/standing_up/video13standing_up06.avi_banked * 2 * 1
Data_test_50-150_Out2/test/sitting_down/video5sitting_down10.avi_banked * 3 * 4
Data_test_50-150_Out2/test/sitting_down/video4sitting_down11.avi_banked * 3 * 1
Data_test_50-150_Out2/test/sitting_down/video5sitting_down14.avi_banked * 3 * 4
Data_test_50-150_Out2/test/sitting_down/video5sitting_down13.avi_banked * 3 * 3
Data_test_50-150_Out2/test/sitting_down/video5sitting_down12.avi_banked * 3 * 1
Data_test_50-150_Out2/test/walking/video9walking39.avi_banked * 4 * 3
Data_test_50-150_Out2/test/walking/video18walking05.avi_banked * 4 * 4
```

Figura C.10: Ejemplo del fichero “svm_resultados”.

```
1 4
1 4
1 4
1 1
1 1
2 4
2 4
2 3
2 1
2 3
2 4
2 5
2 1
3 4
3 1
3 4
3 3
3 1
4 3
4 4
4 4
```

Figura C.11: Ejemplo del fichero “svm_resultados2”.

### C.2.4 Generación de resultados visuales

En este apartado, se muestra la forma de emplear las herramientas de presentación de los resultados de forma gráfica.

Para generar la matriz de confusión que da el nivel de calidad del clasificador entrenado se realiza mediante una aplicación desarrollada en *Matlab*, la cual a partir de los resultados obtenidos en el fichero “svm_resultados2” la genera de manera automática.

En cuanto al la generación del vídeo demostrador es más complejo y se hace mediante dos programas desarrollados en *C++*. El primero de ellos se llama *GetClassesData* y su función es la de obtener los datos en cuanto a posición en el tiempo de las detecciones así como la acción estimada por el clasificador que se está desarrollando. Esto se lleva a cabo mediante la ejecución de la siguiente línea de comando:

```
$ ./GetClassesData svm_resultados
```

Tras esto se debe ejecutar el programa *GetDemo* que es el que genera cada uno de los vídeos del demostrador. Para ello se debe ejecutar la siguiente línea:

```
$ ./GetDemo "Video"
```

Universidad de Alcalá
Escuela Politécnica Superior



ESCUELA POLITECNICA
SUPERIOR

Universidad
de Alcalá